GENERALIZED BLOCKMODELING OF VALUED NETWORKS
(POSPLOŠENO BLOČNO MODELIRANJE OMREŽIJ Z VREDNOSTMI NA POVEZAVAH)

DOKTORSKA DISERTACIJA

LJUBLJANA, 2007
UNIVERZA V LJUBLJANI
FAKULTETA ZA DRUŽBENE VEDE

ALEŠ ŽIBERNA

GENERALIZED BLOCKMODELING OF VALUED NETWORKS
(POSPLOŠENO BLOČNO MODELIRANJE OMREŽIJ Z VREDNOSTMI NA POVEZAVAH)

DOKTORSKA DISERTACIJA

MENTOR: RED. PROF. DR. ANUŠKA FERLIGOJ

LJUBLJANA, 2007
ACKNOWLEDGEMENTS

At the start of this thesis I would like to thank to the people who helped and supported me while writing this thesis.

First I would like to thank my supervisor Professor Anuška Ferligoj for several reasons. She suggested this research topic to me and sparked my interest in blockmodeling and network analysis in general, for which I am very appreciative. I am also very grateful she gave me all the freedom I needed and wanted in my research while always being there when I needed advice, counseling or just someone to talk about my ideas with. I also thank her for carefully reading my thesis, discussing it with me and giving me ideas to improve it.

She also introduced me to Professor Patrick Doreian, to whom I am grateful for reviewing and discussing my thesis and two papers that were written based on the thesis.

Further, I would also like to thank the third person in the trio who developed generalized blockmodeling, Professor Vladimir Batagelj, for discussing with me especially the more technical aspects of this thesis and generalized blockmodeling in general. His comments and experience proved most valuable when designing and implementing algorithms that were developed as part of this thesis. I also thank him and Professor Andrej Mrvar for their support in using Pajek and especially in understanding the algorithms and procedures employed in it by supplying me with the source code of these algorithms and discussing it with me.

Next I would like to thank Professor Douglas R. White, the author of the REGE algorithm. By discussing his algorithms with me he helped me to obtain a much deeper understanding of these algorithms. Without him, the section on indirect approaches to regular equivalence would probably be much shorter and new versions of the REGE algorithm might not be developed. I also thank him for reviewing my paper on regular equivalence (which is also incorporated in the thesis) and for his useful comments.

The next person I wish to thank is Dr. Bojana Lobe, with whom I have shared an office for the last half year. She earned her PhD half a year ago and helped me a lot by sharing her experience with the necessary procedures and formatting of the PhD thesis.

Last but not least, I thank my family and friends. I would especially like to thank my parents. They have always encouraged me to pursue my desires to continue my education and supported me in my studies both emotionally and financially.

At the end, I would like to thank the person who means the most to me, the girl I am due to marry on September 1, Tamara Rape. She has always supported me in my studies, always been available when I needed to explain my problems or ideas to someone and comforted me when I was feeling downhearted due to unexpected problems in my research. Especially in the final months, she was very understanding when I was neglecting her due to my thesis work.
# TABLE OF CONTENT

1  INTRODUCTION ............................................................................................................. 1  
1.1  A short overview ........................................................................................................ 1  
1.2  Structure of the dissertation ........................................................................................ 2  
2  NETWORKS .................................................................................................................... . 4  
2.1  Introduction to networks ............................................................................................. 4  
2.2  Definitions .................................................................................................................. 4  
2.3  Types of networks ...................................................................................................... 5  
2.4  Network analysis ........................................................................................................ 6  
2.5  Conclusion ................................................................................................................ .. 7  
3  INTRODUCTION TO BLOCKMODELING ................................................................... 8  
3.1  Description and purpose of blockmodeling.............................................................. 8  
3.2  Equivalences for binary networks ............................................................................ 11  
3.2.1  Structural equivalence ...................................................................................... 11  
3.2.2  Regular equivalence ......................................................................................... 13  
3.2.3  Other equivalences ........................................................................................... 13  
3.3  Equivalences for valued networks ............................................................................ 14  
3.3.1  Structural equivalence ...................................................................................... 15  
3.3.2  Regular equivalence ......................................................................................... 17  
3.3.3  $f$-regular equivalence ..................................................................................... 18  
3.3.4  Other equivalences ........................................................................................... 20  
3.4  Generalized equivalence ........................................................................................... 20  
3.5  Other criteria for blockmodeling ............................................................................ 21  
3.6  Different approaches to blockmodeling ................................................................... 22  
3.7  Conclusion ................................................................................................................ 23  
4  INDIRECT APPROACHES TO BLOCKMODELING ................................................. 25  
4.1  CONCOR ................................................................................................................. 26  
4.2  Basic description of the indirect approach ............................................................... 26  
4.3  Structural equivalence .............................................................................................. 28  
4.4  Regular equivalence ................................................................................................. 29  
4.4.1  Existing versions (or descriptions) of REGE ................................................... 31  
4.4.2  Suggested modifications of REGE ................................................................. 34  
4.4.3  REGE for valued networks where the tie values are influenced by the ‘size’ of the units .............................................................................................................. 35  
4.4.4  Comparison of versions of REGE .................................................................... 38  
4.5  Structural similarity ................................................................................................. 39  
4.6  Conclusion ................................................................................................................ 41  
5  DIRECT APPROACHES TO BLOCKMODELING ..................................................... 42
7.6.2 Censoring the network at a certain value in valued blockmodeling .......... 90
7.6.3 Adaptation to signed networks ................................................................. 91
7.6.4 Giving more importance to larger inconsistencies in valued blockmodeling ................................................................. 91
7.6.5 Different computation of inconsistencies for regular blocks in homogeneity blockmodeling ................................................................. 91
7.6.6 Possible problems of implicit blockmodeling solutions using null blocks .... 92
7.6.7 Using valued blockmodeling on binary networks ........................................ 93
7.6.8 Using generalized blockmodeling on row and column normalized networks .................................................................................. 93
7.7 Conclusion ........................................................................................................ 93
8 CURRENT IMPLEMENTATION IN R .......................................................................... 98
8.1 Generalized blockmodeling ............................................................................ 98
8.2 Indirect approaches ......................................................................................... 100
8.3 Other functions ............................................................................................... 100
9 APPLICATION AND COMPARISON OF DIFFERENT APPROACHES TO THE BLOCKMODELING OF VALUED NETWORKS ..................................................... 101
9.1 Example 1: Simple artificial example ............................................................... 102
9.1.1 Binary blockmodeling ................................................................................. 103
9.1.2 Valued blockmodeling ............................................................................... 104
9.1.3 Implicit blockmodeling ............................................................................... 106
9.1.4 Homogeneity blockmodeling ..................................................................... 107
9.1.5 Conclusion .................................................................................................. 108
9.2 Example 2: Artificial REGE examples ........................................................... 109
9.3 Example 3: The cooperation of occupational social workers with other organizations ..................................................................................... 112
9.3.1 Indirect approaches .................................................................................... 114
9.3.2 Generalized blockmodeling – structural equivalence ................................. 115
9.3.3 Homogeneity and implicit blockmodeling – regular equivalence ............... 118
9.3.4 Conclusion .................................................................................................. 121
9.4 Example 4: Borrowing notes between social-informatics students .................. 122
9.4.1 Indirect approach using structural equivalence ........................................... 123
9.4.2 Direct approaches using structural equivalence .......................................... 124
9.4.3 Homogeneity blockmodeling according to regular equivalence .................. 125
9.4.4 Implicit blockmodeling according to regular equivalence ......................... 128
9.4.5 Valued blockmodeling according to regular equivalence ......................... 129
9.4.6 Binary blockmodeling according to regular equivalence ............................ 129
9.4.7 REGE ........................................................................................................ 131
11 CONCLUSIONS ........................................................................................................................................................................... 210
11.1 A short overview ............................................................................................................................................................................. 210
11.2 Comparison and evaluation of the proposed approaches ........................................................................................................... 214
  11.2.1 Indirect approaches for regular equivalence ......................................................................................................................... 214
  11.2.2 Generalized blockmodeling versus indirect approaches ......................................................................................................... 215
  11.2.3 Generalized blockmodeling types .............................................................................................................................................. 216
  11.2.4 Conclusions based on comparison of approaches ...................................................................................................................... 219
11.3 Suggestions regarding the usage of approaches to blockmodeling of valued networks ........................................................................................................ 220
  11.3.1 Some suggestions on which approach to use ................................................................................................................................. 220
  11.3.2 Suggested sequence of methods when several are used ........................................................................................................... 222
11.4 Further research ............................................................................................................................................................................... 222
  11.4.1 Improving/testing homogeneity blockmodeling ......................................................................................................................... 223
  11.4.2 Improving implicit blockmodeling .............................................................................................................................................. 223
  11.4.3 Extending generalized blockmodeling to ordinal valued data .................................................................................................... 224
  11.4.4 Better implementation of generalized blockmodeling of valued networks ............................................................................... 224
  11.4.5 New block types ........................................................................................................................................................................... 225
  11.4.6 Using generalized blockmodeling on multi-relational networks ................................................................................................... 225
  11.4.7 Using several generalized blockmodeling types together on the same (one-relational) network or on multi-relational networks ........................................................................................................ 225
  11.4.8 Multi-mode and multi-way blockmodeling .......................................................................................................................... 226
  11.4.9 Missing data and network boundary problem ........................................................................................................................... 227
12 REFERENCES ................................................................................................................................................................................. 229
13 ABSTRACTS ................................................................................................................................................................................ 235
  13.1 English abstract ............................................................................................................................................................................... 243
  13.1.1 Introduction to generalized blockmodeling ................................................................................................................................. 243
  13.1.2 A short overview ........................................................................................................................................................................... 244
  13.2 Slovene abstract (Slovenski povzetek) .............................................................................................................................................. 248
  13.2.1 Kratek uvod v posplošeno bločno modeliranje ................................................................................................................................. 248
  13.2.2 Kratko lok analize disertacije ......................................................................................................................................................... 249
14 POSPLOŠENO BLOČNO MODELIRANJE OMEŽI Z VREDNOSTMI NA POVEZAVAH (RAZŠIRJEN PVOZETEK) ......................................................................................................................... 254
  14.1 Omeži ............................................................................................................................................................................................... 254
  14.2 Uvod v bločno modeliranje ................................................................................................................................................................. 255
  14.3 Bločno modeliranje ............................................................................................................................................................................. 257
  14.3.1 Posredni pristopi ........................................................................................................................................................................... 257
  14.3.2 Neposredni pristopi ....................................................................................................................................................................... 258
  14.4 Posplošeno bločno modeliranje ......................................................................................................................................................... 259
14.4.1 Kriterijska funkcija ................................................................. 260
14.4.2 Binarno bločno modeliranje ....................................................... 261
14.4.3 Obteženo bločno modeliranje ..................................................... 262
14.4.4 Homogeno bločno modeliranje .................................................... 264
14.4.5 Implicitno bločno modeliranje ...................................................... 266
14.4.6 Računanje vrednosti v bločnem modelu ..................................... 268
14.4.7 Predlagane izboljšave, spremembe, alternativne uporabe ................. 269
14.4.8 Implementacija ........................................................................... 270
14.5 Primeri .......................................................................................... 270
14.6 Simulacije ...................................................................................... 272
14.7 Zaključek ....................................................................................... 272
15 APPENDIX ...................................................................................... 275
Appendix A: Example from Section 7.3.3 .......................................... 275
Appendix B: Rules for selecting the dominant row of a column in row- or column-
dominant blocks for computing tie values in reduced graphs based on homogeneity
blockmodeling ..................................................................................... 277
Appendix C: Pairwise comparisons of partitions obtained for the ‘Sweet’ network
(Example 2) with generalized blockmodeling computed using Adjusted Rand Index ..... 279
Appendix D: Figure 9 from Luczkovich et al. (2003) – A graph of the St. Marks food web
and its image graph ............................................................................. 280
Appendix E: Appendix B from Luczkovich et al. (2003) – A list of the identification codes
and compartment names for the St. Marks seagrass carbonflow food web ............ 281
Appendix F: Reduced graph produced by Nordlund (2007, 64 – Fig. 2) presented in a
similar way as the reduced graph by Luczkovich et al. (2003) presented in Appendix D. 282
Appendix G: Results of simulations in Stage 1 for all settings with values of shape1
parameter 10, 6 and 2 ........................................................................ 283
Appendix H: Results of simulations showing the effect of blocks size normalization in
Stage 1 for selected settings with values of shape1 parameter 10, 6 and 2............. 286
Appendix I: Results of simulations showing the effect of block size and maximum
normalization (and their interaction) on implicit blockmodeling in Stage 1 for selected
settings with values of shape1 parameter 10, 6 and 2 .............................. 289
Appendix J: Results of simulations in Stage 2 in tabular form with information on the mean
of the Adjusted Rand Index, its standard error (se), and the number of repetitions (n) used
in its computation ............................................................................. 291
LIST OF FIGURES

Figure 7.1: A network with an optimal partition .................................................................. 76
Figure 7.2: A network with an optimal partition .................................................................. 95
Figure 9.1: Simple artificial network .................................................................................. 103
Figure 9.3: Network 2 ......................................................................................................... 110
Figure 9.4: Network 3 ......................................................................................................... 111
Figure 9.5: Network 4 ......................................................................................................... 112
Figure 9.6: Matrix of the cooperation of occupational social workers with other organizations and a ‘hand-picked’ partition ........................................................................ 113
Figure 9.7: Dendrograms obtained by Ward's hierarchical clustering on (dis)similarities computed using structural equivalence and REGGE (regular equivalence) computed on a valued network ................................................................................................................... 114
Figure 9.8: Dendrograms obtained by Ward's hierarchical clustering on dissimilarities computed according to structural equivalence on a binarized network ............................................. 115
Figure 9.9: Matrices partitioned in terms of structural equivalence ....................................... 116
Figure 9.10: Image graph using structural equivalence for a ‘hand-picked’ partition ...... 117
Figure 9.11: Matrices partitioned using homogeneity and implicit blockmodeling according to regular equivalence ................................................................................................................... 119
Figure 9.12: Image graphs according to the max-regular equivalence for a ‘hand-picked’ partition ................................................................................................................................. 121
Figure 9.13: Valued network of borrowing notes between social-informatics students .... 122
Figure 9.14: Dendrogram obtained by Ward's hierarchical clustering on corrected Euclidean-like dissimilarity (Burt and Minor, 1983 in Batagelj et al., 1992: 71) with p = 2... ........................................................................................................................... 123
Figure 9.15: Matrix partitioned using an indirect approach based on structural equivalence. ............................................................................................................................................... 124
Figure 9.16: Matrix partitioned using homogeneity blockmodeling according to structural equivalence ................................................................................................................................. 124
Figure 9.17: Matrix partitioned using implicit blockmodeling according to structural equivalence .................................................................................................................................. 125
Figure 9.18: Matrix partitioned using valued and binary blockmodeling according to structural equivalence .................................................................................................................................. 126
Figure 9.19: Optimal partitions for sum of squares and absolute deviations homogeneity blockmodeling according to mean-regular equivalence ........................................................................ 126
Figure 9.20: Optimal partitions for homogeneity blockmodeling according to max-regular equivalence ................................................................................................................................... 127
Figure 9.21: Networks of borrowing notes partitioned using implicit blockmodeling according to regular equivalence ........................................................................................................................... 128
Figure 9.22: Optimal partition for valued blockmodeling obtained using null and max-
regular blocks with $m$ equal to 5 or using null and sum-regular blocks with $m$ equal to 10...
Figure 9.23: Histogram of cell values for the matrix in Figure 9.13..................130
Figure 9.24: Optimal partitions for binary blockmodeling according to regular equivalence
with different values of slicing threshold $t$ ..............................................................131
Figure 9.25: Dendrograms based on REGGE similarities.................................132
Figure 9.26: Network of borrowing notes partitioned using different versions of REGE .132
Figure 9.27: Matrix and graphic representation of the ‘sweet’ island from EAT ..........135
Figure 9.28: Dendrogram obtained by Ward's hierarchical clustering on dissimilarities in
terms of structural equivalence computed from the network of the ‘sweet’ island.......135
Figure 9.29: Dendrograms obtained by Ward's hierarchical clustering on (dis)similarities in
terms of regular equivalence computed from the network of the ‘sweet’ island .......136
Figure 9.30: Matrices of the network of the ‘sweet’ island partitioned using REGGE-OW
and REGDI-OW ...........................................................................................................137
Figure 9.31: ‘Optimal’ partition for the binary network of the ‘sweet’ island according to
regular equivalence with a valued and binary representation of the matrix ...........138
Figure 9.32: Histogram of tie values (0s are excluded) in the network of the ‘sweet’ island..
............................................................................................................................. ...139
Figure 9.33: ‘Optimal’ partitions into two clusters of the ‘sweet’ island obtained with
binary blockmodeling according to regular equivalence for networks sliced at different
thresholds (t) ....................................................................................................................... 139
Figure 9.34: ‘Optimal’ partitions into three clusters of the ‘sweet’ island obtained with
binary blockmodeling according to regular equivalence for networks sliced at different
thresholds (t) ....................................................................................................................... 140
Figure 9.35: Matrices of the network of the ‘sweet’ island partitioned into two clusters
using valued blockmodeling according to max-regular equivalence ......................141
Figure 9.36: Matrices of the network of the ‘sweet’ island partitioned into three clusters
using valued blockmodeling according to max-regular equivalence ......................143
Figure 9.37: Distributions of the row sums, column sums and the means of the row and
column sums .......................................................................................................................143
Figure 9.38: Matrices of the network of the ‘sweet’ island partitioned into two clusters
using homogeneity blockmodeling according to f-regular equivalence ....................145
Figure 9.39: Matrices of the network of the ‘sweet’ island partitioned into three clusters
using homogeneity blockmodeling according to f-regular equivalence ....................146
Figure 9.40: Matrices of the network of the ‘sweet’ island partitioned into two clusters
using implicit blockmodeling according to max-regular equivalence ......................147
Figure 9.41: Matrices of the network of the ‘sweet’ island partitioned into three clusters
using implicit blockmodeling according to max-regular equivalence

Figure 9.42: Graphic representation of Adjusted Rand Indices obtained by comparing of two-cluster partitions

Figure 9.43: Row and column normalized matrices of the St. Marks carbon flow network

Figure 9.44: Binarized row and column normalized matrices of the St. Marks carbon flow network

Figure 9.45: Dendrograms obtained by Ward's hierarchical clustering on dissimilarities obtained by deducting REGGE similarities from 1

Figure 9.46: Screeplot for heights of joints obtained by Ward's hierarchical clustering on dissimilarities obtained by deducting REGGE similarities from 1

Figure 9.47: Partitioned normalized matrices obtained using the partitions obtained by Ward's hierarchical clustering on 1 – original REGGE similarities

Figure 9.48: Partitioned normalized matrices obtained using by Ward's hierarchical clustering on 1 – REGGE-NM similarities

Figure 9.49: Image graphs obtained by Nordlund's (2007) method on the partitions obtained by Ward's hierarchical clustering on 1 – original REGGE similarities

Figure 9.50: Image graphs obtained using valued blockmodeling and averaging rules on the partitions obtained by Ward's hierarchical clustering on 1 – original REGGE similarities

Figure 9.51: Image graphs obtained by Nordlund’s (2006) method on the partitions obtained by Ward's hierarchical clustering on 1 – REGGE-NM (modified REGGE) similarities

Figure 10.1: The second density minimum

Figure 10.2: The beta distributions used to generate the ‘0’ ties

Figure 10.3: The beta distributions used to generate the ‘1’ ties

Figure 10.4: Results of simulations for all settings with shape1 = 8

Figure 10.5: Results of simulations for all settings with shape1 = 4

Figure 10.6: Effects of block size normalization for selected settings with shape1 = 8

Figure 10.7: Effects of block size normalization for selected settings with shape1 = 4

Figure 10.8: Effects of block size and maximum normalization (and their interaction) on implicit blockmodeling for selected settings with shape1 = 8

Figure 10.9: Effects of block size and maximum normalization (and their interaction) on implicit blockmodeling for selected settings with shape1 = 4

Figure 10.10: Effects of a different way of simulating the ‘0’ ties
Figure 10.11: Results of simulations in Stage 2 for all settings with shape1 = 8............193
Figure 10.12: Results of simulations in Stage 2 for all settings with shape1 = 4............194
Figure 10.15: Results obtained by a local optimization of 20 and 100 random starting points and of the correct partition..........................................................202
Figure 10.16: Comparison of inconsistencies.................................................................203
Figure 10.17: Results for simulations based on generalized equivalence .................205
Slika 14.1: Omrežje z razbitjem..........................................................268
LIST OF TABLES

Table 4.1: Classification of versions of REGE................................................................. 39
Table 6.1: Schema of ideal and empirical block types..................................................... 49
Table 6.2: Description of ideal blocks for block types (Doreian et al., 2005: 223).............. 50
Table 6.3: Computation of block type inconsistencies for binary blockmodeling (slightly adapted from Doreian et al., 2005: 224) ................................................................ 51
Table 6.4: The rewritten original examples of Batagelj and Ferligoj (2000: 13)............... 52
Table 7.1: Description of ideal blocks............................................................................... 56
Table 7.2: Computation of block type inconsistencies for binary and valued (generalized from binary) blockmodeling .................................................................................. 58
Table 7.3: Descriptions of ideal blocks for homogeneity blockmodeling......................... 62
Table 7.4: Block inconsistencies for homogeneity blockmodeling................................... 63
Table 7.5: Descriptions of ideal blocks for implicit blockmodeling................................. 69
Table 7.6: Computation of block type inconsistencies for implicit blockmodeling without block size normalization ............................................................................................... 70
Table 7.7: Computation of block type inconsistencies for implicit blockmodeling without maximum and block size normalization ........................................................................... 73
Table 7.8: Examples of averaging rules for interval and ordinal networks proposed by Batagelj (1997:151 – Table 4) for block R(X,Y) ............................................................. 84
Table 7.9: Rules for computing tie values in a reduced graph based on homogeneity blockmodeling – general rules................................................................................................. 86
Table 7.10: New null and complete block inconsistencies obtained by combining binary and valued blockmodeling............................................................... 89
Table 8.1: Currently implemented block types by blockmodeling types.......................... 99
Table 9.1: Total inconsistencies of 'optimal' partitions into two and three clusters obtained with binary blockmodeling according to regular equivalence.......................... 140
Table 9.2: Total inconsistencies of 'optimal' partitions into two and three clusters obtained with valued blockmodeling according to max-regular equivalence................. 142
Table 9.3: The values of parameter m when using sum-regular equivalence that resulted in the max-regular partitions of the network of the ‘sweet’ island.................................. 144
Table 9.4: Total inconsistencies of 'optimal' partitions into two and three clusters obtained with valued blockmodeling according to sum-regular equivalence....................... 144
Table 9.5: Total inconsistencies of the 'optimal' partitions into two and three clusters obtained with homogeneity blockmodeling ................................................................. 146
Table 9.6: Total inconsistencies of the 'optimal' partitions into two and three clusters obtained with implicit blockmodeling.................................................................................. 147
Table 10.1: Notation used to compute the Adjusted Rand Index...................................... 174
Table 10.2: Settings used in the simulations – Stage 1 ..................................................... 176
Table 10.3: Values (codes) in Table 10.2 (for Stage 1) .......................................................... 177
Table 10.4: Settings used in the simulations – Stage 2 .......................................................... 191
Table 10.5: Values (codes) in Table 10.4 (for Stage 2) .......................................................... 192
Tabela 14.1: Definicije idealnih blokov za vse vrste bločnega modeliranja ......................... 263
1 INTRODUCTION

The aim of this dissertation is to develop approaches to the generalized blockmodeling of valued networks since so far the generalized blockmodeling approach has only been developed for binary and signed networks by Doreian et al. (2005). In addition, existing approaches that could be used for the blockmodeling of valued networks are discussed and evaluated. For one such approach, an indirect approach blockmodeling according to regular equivalence, some modifications are also suggested. This is also done with the aim to find the best methods or approaches for the blockmodeling of different types of valued networks\(^1\) in the search for optimal solutions with different characteristics\(^2\).

1.1 A short overview

Generalized blockmodeling or more broadly blockmodeling forms part of (social) network analysis. Simply put, a network is a set of units with one or more relations defined on them. Wasserman and Faust (1994: 3-4) stated that network analysis is based on the assumption of the importance of relationships (relations) among the interacting units. Blockmodeling is a method for partitioning the units of a network and determining the pattern of relations among (obtained) clusters. Blockmodeling seeks clusters of equivalent units based on some notion of equivalence. As noted by Doreian (1988a: 243) ‘Equivalence has become a foundational concept in social network analysis’. The two most widely used equivalences are structural and regular equivalence.

For the purpose of this dissertation, one of the most important divisions of blockmodeling approaches is into indirect and direct approaches. Indirect approaches first compute some measure of similarity or dissimilarity among the units of a network based on a selected measure of equivalence and then use one of the classical clustering techniques to uncover clusters of units, while direct approaches directly search for a partition that best fits the selected equivalence as measured by a selected criterion function (Batagelj et al., 1992b: 66).

Generalized blockmodeling is based on the direct approach. When compared to other direct approaches, its main strength is its adaptability. It can be used to perform blockmodeling according to different types of equivalences, including generalized equivalence. Generalized equivalence is not a specific type of equivalence but more of a concept for building ‘custom’ equivalences. It is defined by specifying allowed types of connections between and within clusters. However, up till now generalized blockmodeling has only been developed for binary and signed networks.

\(^1\) Which kinds of different networks were considered is set out in Chapter 2 (Networks).
\(^2\) The different characteristics of the solutions mainly reflect the type of equivalence sought. More on this can be found in Sections 3.2 and 3.3.
In this dissertation new approaches to the generalized blockmodeling of valued networks are developed. However, as they can still be regarded as approaches to generalized blockmodeling as presented by Doreian et al. (2005) the same type of criterion function can be used. The most important differences between the approaches to generalized blockmodeling presented by Doreian et al. (2005) and those developed here is that the approaches presented by Doreian et al. (2005) can be applied to binary and signed networks, while those presented in this dissertation can be applied to valued networks. To achieve that, new block types appropriate for valued networks are introduced. The common characteristics of all approaches to generalized blockmodeling (including those developed here) are, in addition to the common basic criterion function, their ability to specify the desired solution either by a type of equivalence (which is then translated into allowed block types) or by generalized equivalence. Generalized equivalence is defined directly by the allowed block types or even more precisely by the desired blockmodel.

In addition to generalized blockmodeling, other approaches to blockmodeling are reviewed (indirect blockmodeling and other direct approaches). For one family of algorithms that falls into the category of indirect approaches, the REGE algorithms for computing (dis)similarities in terms of regular equivalence, some modified versions are also presented.

All suggested approaches are implemented in a blockmodeling package (Žiberna, 2007a) for statistical program and environment R (R Development Core Team 2006). They and some other approaches are then tested on several empirical and constructed examples. A relatively comprehensive simulations study is also conducted where the proposed approaches are evaluated on valued regular networks. The proposed approaches are compared based on their theoretical characteristics and their performance in the examples and simulations.

### 1.2 Structure of the dissertation

The dissertation is split into three parts. The first part includes Chapters 1 to 6. These chapters introduce the topic of the dissertation and give an overview of the relevant literature, although some contribution to the scientific field is also made. Chapter 7 is the main theoretical chapter. Here new approaches to the generalized blockmodeling of valued networks are proposed. In Chapters 8 to 11, these approaches are implemented and evaluated (together with some others) on empirical and artificial examples and through simulations. In Chapter 11, the evaluation is completed based on the findings of all previous chapters.

In this first chapter, the aim, overview and structure of the thesis are presented. In the next chapter the networks, the data structures that are analyzed by the proposed approaches, are introduced, together with the field devoted to studying them (network analysis). In Chapter 3, blockmodeling is introduced. Equivalences as the central concept in blockmodeling are reviewed. In addition, regular equivalence especially its extensions to valued networks are

---

3 Where values on the ties of relations are measured at least on an interval scale.
discussed in-depth. A new $(f\text{-regular})$ equivalence for valued networks, comparable to regular equivalence, is suggested. At the end of Chapter 3, different approaches to blockmodeling that are presented in more detail in the following two chapters are introduced. In Chapter 4 the indirect approach to blockmodeling is presented. In addition to a review of the literature, several new versions of one of the algorithms used in the direct approach, REGE, are suggested. In Chapter 5 the direct approaches are reviewed. The most important of these, generalized blockmodeling, is presented in Chapter 6. The current state of generalized blockmodeling is reported here as the basis for the new approaches presented in the next chapter. Most of it applies to binary networks, although some suggestions for valued networks are also suggested.

Chapter 7 is the core chapter of the thesis and is therefore presented more thoroughly. Here, approaches to the generalized blockmodeling of valued networks are developed. These approaches are valued blockmodeling, homogeneity blockmodeling and implicit blockmodeling. Valued blockmodeling is developed in Section 7.1 and homogeneity blockmodeling in Section 7.2. The ideas of Batagelj and Ferligoj (2000: 11-13) presented in the previous chapter are further developed in implicit blockmodeling in Section 7.3. In Section 7.4 the null block problem, which affects homogeneity and in some cases implicit blockmodeling, is discussed and some solutions to it are suggested. In Section 7.5, averaging rules, that is the rules for computing tie values in reduced graphs, are presented. First the ideas of Batagelj (1997: 148, 151) are outlined and then for each presented generalized blockmodeling approach suitable rules are suggested. The slightly modified rules suggested for one version of homogeneity blockmodeling are argued to be suitable as general rules for computing tie values in reduced graphs. In Section 7.6, some improvements to the suggested approaches, alternative versions of them and additional uses of these approaches are presented. The chapter concludes with a short review and comparison of generalized blockmodeling approaches.

The remaining chapters (Chapters 8 to 11) deal with the application and evaluation of the proposed approaches. In Chapter 8, the implementation of these approaches in a blockmodeling package (Žiberna, 2007a) for statistical program and environment R (R Development Core Team 2006) is described. This software is used to apply and evaluate the proposed approaches in empirical and artificial valued networks (Chapter 9). In Chapter 10, a simulation study to further evaluate the proposed approaches to generated valued regular networks is presented. In the last chapter, Chapter 11, all proposed approaches are evaluated and compared from both theoretical and practical perspectives (results from Chapters 9 and 10). In addition, an overview of the whole thesis is provided and the scientific contributions are emphasized.
2 NETWORKS

In this chapter networks and network analysis are introduced. Networks are data structures that are analyzed by the blockmodeling approaches suggested and reviewed in this thesis. These and several other methods are used in the field of network analysis.

In this chapter, first a quick introduction to networks is given. This is followed by the definition of networks and a description of their different types. The chapter concludes with a description of network analysis, the common name for a number of methodological tools for analyzing networks, of which blockmodeling forms part. Blockmodeling is the focus of this thesis and is introduced in the next chapter.

2.1 Introduction to networks

Networks can be found practically everywhere. They can range from small sociograms such as those introduced by Moreno (1934) with only a few units to huge networks with practically billions of units, for example a network of all computers connected to the Internet or a network of all people and their acquaintances. We can form networks wherever we can find units that are connected in some way with each other. A more formal definition is given in the next subsection. Units can be people, organizations, countries, words etc. and for each of these types of units there are a number of possible ties between pairs of these units. Commonly analyzed networks are networks among people (friendship, communication, help etc.), trade networks among organizations, countries, citation networks, genealogies, organic molecules in chemistry, ties among words in text, transportation networks etc. Examples of visualizations, with some even dating back to Roman times while others are modern, can also be found on slides presented by Batagelj and Ferligoj (2003: 1-11). A large collection of networks in a form appropriate for analysis with common network analysis programs can be found on a web page of Pajek datasets maintained by Batagelj (2003).

The networks analyzed in this dissertation may all be considered as small. The methods developed here are computationally intensive and are incapable of analyzing large networks.

2.2 Definitions

‘A social network consists of a finite set of actors and the relation or relations defined on them’ (Wasserman and Faust, 1994: 20). The above definition can also be used for non-social networks. Since the term actor (in the definition by Wasserman and Faust) is specific to social networks, the term unit is used instead in this dissertation. There are a few exceptions where other terms are used, such as actor, vertex, point etc. that only appear in direct citations. All of these terms label the same concept. Often in such cases the term unit is added in brackets. This practice is also used in other cases where a different term is used in a direct citation than is usually used in this dissertation.
The network can be written as \( N = (U, R_1, R_2, \ldots, R_r) \), where \( U \) is a set of all units \( U = (u_1, u_2, \ldots, u_n) \) and \( R_t \) is the relation between these units \( R_t \subseteq U \times U \). Mathematically speaking, a network can be represented as a multigraph, a graph with multiple relations (White and Reitz, 1983: 203) or as a collection of graphs or digraphs (or both) with a common vertex set (set of units) (Everett and Borgatti, 1993: 243). If a network has only one relation, the representation is a graph. The relations in a graph are represented by either arcs, edges or sometimes both. The arcs represent directed relations and the edges represent undirected relationships. A graph with arcs is also called a directed graph or digraph.

In generalized blockmodeling, a graph or one relation in a network is usually represented by a matrix \( R \) with elements \( r_{ij} \), where the value of \( r_{ij} \) indicates the value (or weight) of the arc from unit \( i \) to unit \( j \). Networks with multiple relations are represented by several matrices, one for each relation, that can also be combined in a three-dimensional array, called a relational box by Winship and Mandel (1983: 325-326).

De Nooy et al. (2004: 6-7) first define a graph as a set of vertices (units) and a set of lines (relations) between pairs of vertices (units) and, based on this, a network as a graph combined with additional information on the vertices (units) or lines (relations).

In this dissertation, most of the networks only have one relation \( R \) defined on the units, although the methods and tools introduced here could easily be used (in a slightly modified form) on networks involving several relations. The relation \( R \) is usually described by the corresponding valued matrix \( R = [r_{ij}]_{m \times n} \). \( r_{ij} = 0 \) indicates the absence of the relation and all other values indicate the presence of the relation, its strength and direction (sign). In the previously presented approaches to generalized blockmodeling, the values of \( r_{ij} \) were limited to \( \{0, 1\} \) or at most to \( \{-1, 0, 1\} \) or, to put it differently, the tie values (except for those present or absent) was not taken into account. The aim of this dissertation is to extend the generalized blockmodeling approach to valued networks. The developed approaches assume that the relations are measured on at least an interval scale.

### 2.3 Types of networks

Networks can be classified into several types based on different criteria. Some ways of classifying were implicitly already mentioned above. One of the bases of a classification can be the set of possible values that the relations can take or measurement scale. We can differentiate among binary, signed, categorical, valued (all, only nonnegative) and maybe also some other types of networks. Since different relations in a single network might be measured differently, this is more a classification of relations than of networks. As mentioned, this classification is important since many approaches are only applicable to networks that are measured with a certain scale or several (but not all) scales. The methods that are developed in this thesis are designed for (interval) valued networks.

Another classification of networks could be based on the number of different relationships they contain. Although most approaches to blockmodeling focus on networks with only one relation (including the approaches developed here and generalized blockmodeling), they can
usually also be applied to networks with several relations. However, many implementations of these approaches are restricted to only one relation (some procedures in Ucinet 5 (Borgatti, Everett, and Freeman, 1999), Pajek (Batagelj and Mrvar, 2005a, 2005b), and Model2 (Batagelj, 1996)).

Another of the more important classifications is based on the number of distinct sets of units in the networks. This defines the mode of the network. If all units are from the same set the network is referred to as being one-mode, if there are two sets then it is two-mode and so on. In two-mode networks usually not all the units can initiate ties, meaning that all arcs usually lead from one set to another (Wasserman and Faust, 1994: 29-30). More than two-mode networks are quite rare. This dissertation focuses on one-mode networks.

The size (the number of units) of a network can also be used for classification. Only small networks (of up to 50 units) are considered in this thesis since the methods developed in it are computationally too intensive to be applied to larger networks.

### 2.4 Network analysis

One of the descriptions of network analysis might be that it is a collection of models and methods for analyzing (social) networks, as Wasserman and Faust (1994: 3) described the focus of their book entitled ‘Social network analysis: Methods and applications’. While they focused on social networks and talked about social network analysis, the methods they described are not limited to social networks and they even sometimes use the term ‘network analysis’. Since this work is not limited to social networks, I use more the general terms of networks and network analysis. Wellman and Berkowitz (1988: 2-4) also pointed out that should not be reduced to ‘a mystifying jargon or an arcane bag of methodological tricks’. They also clarified the relationship between network analysis and structural analysis.

Wasserman and Faust (1994: 3-4) stated that network analysis is based on the assumption of the importance of relationships (relations) among the interacting units. They also provided four principles that distinguish network analysis from similar approaches (Wasserman and Faust, 1994: 4):

- Actors (units) and their actions (relations) are viewed as interdependent rather than independent, and autonomous units
- Relational ties (linkages) between actors (units) are channels for transfer or ‘flow’ of resources (either material or nonmaterial)
- Network models focusing on individuals (individual units) view the network structural environment as providing opportunities for or constraints on individual action
- Network models conceptualize structure (social, economic, political, and so forth) as lasting patterns of relations among the actors (units)

De Nooy et al. (2004: 5) present the detecting and interpreting of patterns of (social) relations between units as being the main goal of (social) network analysis. Their definition seems
narrower than that offered by Wasserman and Faust. However, the topic of this dissertation (generalized blockmodeling) fits into them both.

The book by Wasserman and Faust (1994) is one of the most widely used textbooks for (social) network analysis and provides a good overview of the field, although due to its age it does not include some of the newer methods. These can, for example, be found in Carrington et al. (2005). Other books on social networks methods are, among others, by Degenne and Forse (1999) and Scott (2002).

The book by De Nooy et al. (2004) focuses of exploratory (social) network analysis, especially on practical issues of using the Pajek program for that purpose, as indicated in the book’s title ‘Exploratory Social Network Analysis with Pajek’.

Blockmodeling is one of the methodological tools of network analysis which is the focus of this thesis. It is introduced in the next chapter.

2.5 Conclusion

In this chapter, several important concepts are introduced. The first of them is a network. A network is a set of units and one or more relations defined on them. Some notation is also introduced.

Several types of networks are distinguished. The most important one for the purposes of this thesis is based on the measurement scale as methods for (interval) valued networks are developed in it. Networks can also be distinguished based on the number of units, number of relations, number of distinct sets of units (modes) in networks etc. Most methods developed in this thesis can only be applied to small networks. Although most can be applied to one or more mode networks and one or more relational networks, they are not used in this thesis. The chapter concludes with a description of network analysis, the field and the collection of methods for studying networks. Blockmodeling is the method of network analysis that is in the focus of this thesis and is covered in the following chapters.
3 INTRODUCTION TO BLOCKMODELING

Blockmodeling is one of the methods used for network analysis. The networks and network analysis are introduced in the previous chapter. In this chapter blockmodeling is introduced. First, a description of blockmodeling is provided and its purpose explained. This is followed by three sections devoted to an introduction to and description of different kinds of equivalences. Section 3.2 reviews equivalences for binary networks. In Section 3.3 their extensions to valued networks are examined.

Generalized equivalence is presented in Section 3.4. Generalized equivalence is not a specific type of equivalence but more of a concept for building ‘custom’ equivalences. It is closely related to generalized blockmodeling, the main topic of this thesis.

After that, other criteria (in addition to formal definitions of equivalences) for blockmodeling are reviewed. The chapter finishes with a review of the different approaches to blockmodeling, some of which are then covered in the following chapters.

3.1 Description and purpose of blockmodeling

‘Blockmodeling tools were developed to partition network actors (units) into clusters, called positions, and, at the same time, to partition the set of ties into blocks that are defined by the positions (see Lorrain and White (1971), Breiger et al. (1975), Burt (1976) for the foundational statements)’ (Doreian et al., 2004: 29). It may be said that blockmodeling seeks clusters of equivalent units based on some notion of equivalence.

A similar definition or description was supplied by Batagelj (1997: 143) who stated that blockmodeling has two basic subproblems:

1. partitioning of units or determining the classes (clusters) that form the vertices in a model; and
2. determining the ties in a model (and their values).

While both definitions are similar, the second definition more clearly emphasizes the need to determine the ties (presence and possible types) and possibly their values between and within clusters or positions in the blockmodel4.

I started with these definitions since unlike some definitions, especially earlier ones, they are general enough to include not only conventional blockmodeling5 but also generalized blockmodeling. For example, the definition of a blockmodel by Wasserman and Faust (1994: 395) stated that it consists of two things:

1. a partition of units in the network in discrete subsets called positions (clusters); and

---

4 Although Batagelj in the above definition does not directly specify the need to determine the type of the tie, it is clear from the paper that he does think about the types and values of the ties in the blockmodel.

5 A definition of the term ‘conventional blockmodeling’ is provided later in the dissertation.
2. for each pair of clusters a statement of the presence or absence of a tie within or between the clusters on each relation.

While the first part of their definition is similar to that given by Batagelj, the second part differs slightly, yet importantly. They only talk about the need to determine the presence or absence of a tie. This definition is only appropriate for conventional blockmodeling and for other approaches to blockmodeling that rest on just one type of equivalence (excluding generalized equivalence) and, even with this restriction, it is not well suited for valued networks.

Borgatti and Everett (1992b: 91) stated that ‘Viewed as a method of data reduction, blockmodeling is a valuable technique in which redundant elements in an observed system are reduced to yield a simplified model of relationships among types of elements (units)’. Here the aim is to combine more or less identical units and thus reduce the complexity or size of the network.

However, they also pointed out that blockmodeling is usually seen ‘as a tool for discovering roles and positions occupied by actors (units) in a social structure (Knoke and Kuklinski, 1982)’ (Borgatti and Everett, 1992b: 92). This is one of the most important uses of blockmodeling. As they pointed out, the problem arises when researches use blockmodeling based on structural equivalence for this purpose, which can only be used as a method of data reduction. This has happened quite frequently, especially since in the past blockmodeling was often thought of only in the sense of structural equivalence. The focus on structural equivalence is clearly seen in the statement by White (1988: 229): ‘Blockmodeling ultimately rests on the concepts of structural equivalence’. They (Borgatti and Everett, 1992b: 92) understood positions as a set of actors that have the same ties with the same types of others and the relationships that these positions have as roles.

Winship and Mandel (1983: 315-317) also stated that blockmodeling is only capable of uncovering positions and not roles in a social network. However, it is clear that they restricted blockmodeling to structural equivalence. In their paper they proposed an alternative (non-blockmodeling) approach to finding roles based on role sets. They actually defined position as a cluster of structurally equivalent units. On the other hand, they associated roles with particular patterns of relations.

Wellman (1988: 40) also stated that ‘… blockmodeling inductively uncovers underlying role structures in a social structure …’. Citing other authors, he also stated that blockmodeling helps to compare actual networks with hypothesized structures. For this purpose, the pre-specified blockmodeling introduced by Batagelj, Ferligoj, and Doreian (1998) is the most appropriate.

Wasserman and Faust (1994: 348-351) cited Linton's (1939: 113-114) definition of social position (he used the term ‘status’) ‘as the polar position in … patterns of reciprocal

6 Types of equivalence are described in the following two sections.
behavior’. They also cited him, noting that when a person ‘puts the rights and duties which constitute the status (position) into effect, he is performing a role’. Wasserman and Faust (1994: 348) stated that:

In social network analysis *position* refers to a collection of individuals who are similarly embedded in networks of relations, while *role* refers to the pattern of relations which obtain between actors or between positions. The notion of position thus refers to a collection of actors who are similar in social activity, ties, or interactions, with respect to actors in other positions.

They continued (Wasserman and Faust, 1994: 349):

The notion of social role is conceptually, theoretically, and formally dependent on the notion of social position. Whereas network position refers to a collection of actors, network role refers to associations among relations that link social positions. Thus, role is defined in terms of collections of relations and the associations among relations. … It is also important to note that roles are defined not simply on the linkages between two positions, but on how relations link the entire collection of actors and positions throughout the network. Thus, social roles can be modeled at three different levels: actors, subset of actors, and the network as a whole.

As Nadel (1957) and Lorrain and White (1971) have observed, role is not just a theoretical construct invented by social scientist, but also can be expressed in our everyday language.

Different authors use the terms positions and roles in a different way, as observed for example by Winship and Mandel (1983: 314-315) and Wasserman and Faust (1994: 348).

Some additional notations are necessary upon the introduction of blockmodeling. The notation of networks has already been introduced and is only repeated here. In addition, a notation for a block, the building block of a blockmodel is added. The notation used is:

- The network N = (U, R), where U is a set of all units U = (u₁, u₂, ..., uₙ) and R is the relation between these units R ⊆ U x U. (The network can also have multiple relations N = (U, R₁, R₂, ..., Rₘ), where m is the number of relations.
- Relation R can also be presented by the matrix R = [rᵢⱼ]ₙₓₙ, where rᵢⱼ is the value (or weight) of an arc (or edge in undirected graphs) from unit i (or uᵢ) to unit j (or uⱼ).
- C = {C₁, C₂, ..., Cₖ} is a partition (or clustering) of the set U in K clusters. Φ is a set of all feasible clusterings. A clustering C also partitions the relation R into blocks. R(Cᵢ, Cⱼ) = R ∩ Cᵢ x Cⱼ. Each such block consists of units belonging to clusters Cᵢ and Cⱼ and all arcs leading from cluster Cᵢ to cluster Cⱼ. If i = j, the block R(Cᵢ, Cᵢ) is called a diagonal block.
- nᵢ – the number of units in cluster Cᵢ.
3.2 Equivalences for binary networks

As stated previously, blockmodeling seeks clusters of equivalent units based on some notion of equivalence.

Equivalence has become a foundational concept in social network analysis. Each use of an equivalence concept has two components: (i) the definition of equivalence and (ii) a computational algorithm for detecting equivalences, or the extent to which they exist. (Doreian, 1988a: 243)

In this and the following two sections (Sections 3.2, 3.3 and 3.4), the definitions of equivalences are discussed. The algorithms are discussed in the following chapters, namely in Chapters 4 to 7.

Several types of equivalences are used in social network analysis. The most widely used are structural and regular equivalence. Other equivalences include automorphic equivalence and perfect equivalence. The relationships among them are described in Everett and Borgatti (1994). In this subsection, the most commonly used equivalences are defined for binary networks. This is in fact how they were defined originally and they also provide a starting point for possible definitions of equivalences for valued networks. The definitions or possible definitions for valued networks are presented in Section 3.3.

Doreian et al. (1994) introduced the concept of generalized equivalence. Generalized equivalence is defined by a set (and optionally positions) of allowed block types. Most of these equivalences are presented in more detail below (in the following subsections).

3.2.1 Structural equivalence

The definition of structural equivalence was originally given by Lorrain and White (1971) and is presented below. A simple description of the structural equivalence given by White and Reitz (1983: 200) is: ‘Structurally equivalent points (units) are related in the same way to each other and all other point (units)’. However, this definition is not the only definition in existence. Borgatti and Everett (1992a: 5-10) showed there are a number of slightly different definitions of structural equivalence.

The original definition of structural equivalence given by Lorain and White (1971: 63) follows:

Objects \( a, b \) of category \( C \) are structurally equivalent if, for any morphism \( M \) and any object \( x \) of \( C \), \( aMx \) if and only if \( bMx \), and \( xMa \) if and only if \( xMb \). In other words, \( a \) is structurally equivalent to \( b \) if \( a \) relates to every object \( x \) of \( C \) in exactly the same way as \( b \) does. From the point of view of the logic of the structure, \( a \) and \( b \) are absolutely equivalent, they are substitutable.

---

7 A more formal definition by White and Reitz (1983) is also given below.
Borgatti and Everett (1992a: 6) also noted that a different definition is in use today, one presented by Burt (1976: 96), who defined a set of structurally equivalent nodes (units) as a set of nodes (units) connected by the same relations to exactly the same people. They also pointed out a shortcoming of these two definitions. They said that the problem of these two definitions is that, in a graph without loops, people who are connected to each other cannot occupy the same position. This is of course true since both definitions do not recognize the special role of loops and ties between units that are checked for structural equivalence.

Borgatti and Everett (1992a: 7) stated that the definition that works in all cases was given by Everett et al. (1990: 164): ‘Suppose \( G \) is a labeled graph with vertex set \( V \) and edge set \( E \). Then two vertices \( a, b \in V \) are structurally equivalent iff\(^8\) the permutation \( (a \ b) \) produces an automorphism of \( G \).’ In this definition, we can also replace edge set \( E \) by line set \( L \) that incorporates edge set \( E \) and arc set \( A \).

The following definition was presented by White and Reitz (1983: 199):

If \( G = (V, R) \)\(^9\) and \( \equiv \) is an equivalence relation on \( V \) then \( \equiv \) is a structural equivalence if and only if for all \( a, b, c \in V \) such that \( a \neq b \neq c \), \( a \equiv b \) implies:

(i) \( aRb \) if and only if \( bRa \);
(ii) \( aRc \) if and only if \( b Rc \);
(iii) \( cRa \) if and only if \( cRb \); and
(iv) \( aRa \) implies \( bRb \).

Structurally equivalent points (units) are related in the same way to each other and to all other points (units).

This definition has a typing error in it, namely, the last point should read ‘\( aRa \) implies \( bRb \)’.

Having this typing error in mind, the same definition was presented by Batagelj et al. (1992b: 66) with a different notation. Here it is presented using the same notation (and form) as in White and Reitz (1983):

If \( G = (V, R) \)\(^10\) and \( \equiv \) is an equivalence relation on \( V \) then \( \equiv \) is a structural equivalence if and only if for all \( a, b, c \in V \) such that \( a \neq c \) and \( b \neq c \), \( a \equiv b \) implies:

(i) \( aRb \) if and only if \( bRa \);
(ii) \( aRc \) if and only if \( b Rc \);
(iii) \( cRa \) if and only if \( cRb \); and
(iv) \( aRa \) implies \( bRb \).

This definition is used as a definition of structural equivalence in this text.

---

\(^8\) This is standard notation for \( if \ and \ only \ if \). In this text, the longer version is used.

\(^9\) \( V \) is used instead of \( P \) in the original definition to represent the vertex set in an attempt to use common notation in the dissertation. When referring to networks instead of graphs, \( U \) as a unit set is used in its place.

\(^10\) \( V \) is used instead of \( P \) in the original definition to represent the vertex set in an attempt to use common notation in the dissertation. When referring to networks instead of graphs, \( U \) as a unit set is used in its place.
As already mentioned in the Section 3.2.1, structural equivalence is generally not suited for discovering roles and positions in social networks, although it has been used for that purpose (Borgatti and Everett, 1992b: 91; Winship and Mandel, 1983: 315). However, it is well suited for combining (nearly) identical units in a network and data reduction (Borgatti and Everett, 1992b: 91). Of course, we note that the terms roles and positions have not been used consistently in the literature. Here, we adopt the meaning that corresponds to Wasserman and Faust's (1994: 348-351) use of social network analysis positions and roles. For example, (Winship and Mandel, 1983: 315) defined position as a cluster of structurally equivalent units and that for finding such positions structural equivalence is obviously well suited.

3.2.2 Regular equivalence

The other common type of equivalence is regular equivalence. ‘Regularly equivalent points (units) are connected in the same way to matching equivalents’ (White and Reitz, 1983: 200). A more formal definition is (White and Reitz, 1983: 200):

If \( G = (V, R) \)\(^{11} \) and \( \equiv \) is an equivalence relation on \( V \) then \( \equiv \) is a regular equivalence if and only if for all \( a, b, c \in V \), \( a \equiv b \) implies:

(i) \( aRc \) implies there exists \( d \in V \) such that \( bRd \) and \( d \equiv c \);
(ii) \( cRa \) implies there exists \( d \in V \) such that \( dRb \) and \( d \equiv c \).

Batagelj et al. (1992a: 125) proposed that if \( C = \{C_i\} \) is a partition corresponding to a regular equivalence, then \( R(C_u, C_v) \) is either null (empty) or it has a property whereby there exists at least one 1 (tie) in each of its rows and in each of its columns.

Yet the answer to the question of whether this fully complies with the above definition by White and Reitz (1983: 200) depends on the interpretation of this definition. Obviously, Batagelj et al. (1992a: 125) and Borgatti and Everett (1992b: 102) understood it in a way that allows \( d \) in (i) and (ii) to be different units, while examining the REGGE algorithm by White (1985a) shows that White intended that \( d \) represent the same unit in both (i) and (ii). The interpretation of Batagelj et al. (1992a: 125) and Borgatti and Everett (1992b: 102) is more suitable for the generalized blockmodeling approach because in this way each block can be evaluated independently. Therefore, this one is primarily used in this text.

3.2.3 Other equivalences

Another very common equivalence is automorphic equivalence that was introduced by Winship (1974 in Winship and Mandel, 1983: 324) in an unpublished paper. His definition is: ‘Two individuals (units) \( i \) and \( j \) are automorphically equivalent if there is an automorphism \( f \) such that \( f(i) = j \). The equivalence is also referred to as structural isomorphism (Borgatti and Everett, 1992a: 10).

\(^{11} \) \( V \) is used instead of \( P \) in the original definition to represent the vertex set in an attempt to use common notation in the dissertation. When referring to networks instead of graphs, \( U \) as a unit set is used in its place.
Everett and Borgatti (1994: 40) also pointed out that automorphic equivalence is a natural generalization of (weak)\textsuperscript{12} structural equivalence. While two units that are structurally equivalent cannot be distinguished in a labeled network (if their labels are removed), two units that are automorphically equivalent cannot be distinguished in an unlabelled network.

Everett and Borgatti (1994) also presented a number of other equivalences and colorations\textsuperscript{13} and showed that most of them are special cases of regular equivalence or coloration (exception is ecological coloration). They further developed exact colorations especially in combination with perfect and ecological colorations in Everett and Borgatti (1996). Most of them are not discussed here. An exception is exact coloration\textsuperscript{14} since this one is useful for generalized blockmodeling.

Everett and Borgatti (1994: 43) proposed the following definition: ‘A coloration is exact if whenever two vertices (units) are coloured the same, their neighbourhoods contain not only the same sets of colours, but also the same number of vertices of each colour present’. They also provided a more formal definition using k-colored out- and in-neighborhoods. From the description of equitable partition in Stadler and Tinhofer (1999: 224-226), it can be seen that this is precisely the same as exact coloration.

Everett and Borgatti (1994: 46) also suggested that most of these colorations can also be weakened by demanding that the condition holds only for in- or out-neighborhoods.

Several other equivalences have been proposed. For example, there are role-set equivalence (Winship and Mandel, 1983: 324), (general) local roles (‘local’ or ‘k-local’ versions of several equivalences, including automorphic and regular equivalence) (Everett et al., 1990) and other types. However, these equivalences exceed the scope of this dissertation as they are not suitable for generalized blockmodeling in the sense of partitioning a matrix of a relation (or several matrices of relations) into blocks that exhibit some desired properties (patterns).

3.3 Equivalences for valued networks

There are two main problems with the definitions and descriptions of equivalences presented in the previous section. First, many of them are only defined for binary networks. Sometimes the generalization to valued (especially continuous) networks is straightforward and has already been presented. The simplest generalization, which is also often the most suitable, is to strengthen the definition in such a way that two ties can satisfy the requirement only if they have equal values. This approach is very suitable for structural equivalence yet can be very problematic for some others if the tie values are continuous and not discrete.

\textsuperscript{12} Authors distinguish between week and strong structural equivalence. Here, I put ‘weak’ in brackets since their definition of weak structural equivalence corresponds to the definition of structural equivalence used in this text.

\textsuperscript{13} The authors (Everett and Borgatti, 1994) use the term coloration for the assignment of colors to units. Any equivalence induces coloration and any coloration induces equivalence.

\textsuperscript{14} Everett and Borgatti used the term ‘exact coloration’ in their paper published in 1994 and the term ‘exact regular coloration’ in a paper published in 1996.
The second problem, which closely relates to the previous one, is that some of these equivalences have not yet been operationalized, meaning that algorithms for computing the deviation of a partition from selected equivalence have yet to be developed. This deviation should be zero if and only if the partition perfectly matches the selected equivalence. For binary networks this problem is true only for certain equivalences and certain approaches to blockmodeling. For valued networks, this problem closely relates to the previous. Before a definition (at least implicitly) for valued networks is given the operationalization is impossible.

Nevertheless, some of the equivalences are operationalized for valued networks, meaning that there are algorithms for computing them, although they do not have a clear definition for valued networks. One example is regular equivalence and REGE algorithms (White, 2005).

Another very important question is which requirements must a generalization of an equivalence meet to be considered a good generalization. At least two such requirements are desired:

1. the binary version of the equivalence is only a special case of the valued version that we get if we assume that in binary networks all values on ties are equal to 1; and
2. the definition captures the ‘essence’ of the equivalence.

The first of these two requirements is both essential and can be easily tested. The second one is also very important yet it is difficult to test.

There is a further requirement that must be met, however it was not listed there since it might be more a requirement for the algorithm than a requirement for the definition. This requirement is that the definition or at least the algorithm that searches and tests for the presence of equivalence must take into account the nature of the values, namely, it must not treat them only as nominal.

In this section, the focus is on definitions and not on operationalizations. The operationalizations are mainly only given for generalized blockmodeling in Chapter 7.

3.3.1 Structural equivalence

Structural equivalence is also well-defined and operationalized for continuous valued networks (see Batagelj et al., 1992b: 65-66; Borgatti and Everett, 1992b: 101; Wasserman and Faust, 1994: 356-360; Batagelj and Ferligoj, 2000: 12-13; Breiger and Mohr, 2004: 21-24; and others). Both indirect and direct approaches can be used for valued networks. Structural equivalence for valued networks means that two units are equivalent if they are connected with arcs of equal values to the rest of the network.

\[15\] For example, automorphic and perfect equivalences have not been incorporated in generalized blockmodeling, meaning that an appropriate set of ideal blocks has not yet been proposed. For some equivalences this is also impossible in the context of generalized blockmodeling.
Borgatti and Everett (1992b: 98 – Definition 5) gave a formal definition of structural equivalence for valued networks. However, their definition can also be rewritten as follows (in a simplified form for two-way, one-mode networks to make it more comparable with some other definitions presented below): suppose that \( \equiv \) is an equivalence relation on \( U \) then \( \equiv \) is a structural equivalence if and only if for all \( a, b \in U, a \equiv b \) implies:

1. \( r_{bi} = r_{ai} \) for all \( i \in U \),
2. \( r_{ib} = r_{ia} \) for all \( i \in U \),

However, this definition of structural equivalence has exactly the shortcomings Borgatti and Everett (1992b, 5-10) criticized in the original definition of structural equivalence given by Lorrain and White (1971). These shortcomings are shown in the fact that according to this definition in a network without loops two units that are connected cannot be structurally equivalent. This happens because the loops and ties between the units that are checked for structural equivalence are not treated any differently to any other cells in the matrix.

A definition overcoming these shortcomings was given by Batagelj et al. (1992b: 66): suppose that \( \equiv \) is an equivalence relation on \( U \) then \( \equiv \) is a structural equivalence if and only if for all \( a, b \in U, a \equiv b \) implies:

1. \( r_{bi} = r_{ai} \) for all \( i \in U \setminus \{a, b\} \),
2. \( r_{ib} = r_{ia} \) for all \( i \in U \setminus \{a, b\} \),
3. \( r_{bb} = r_{aa} \), and
4. \( r_{ab} = r_{ba} \).

Here, the special status of the loops and ties between the two units that are being evaluated for structural equivalence is taken into account by treating them separately by excluding them from conditions 1 and 2 and by adding two conditions that correspond to these special cases. This definition corresponds to the definition of structural equivalence that is used in this text.

Based on implementations (the ‘ignore’ method for handling diagonal values in Ucinet 5 (Borgatti et al., 1999)) and the dissimilarities (Batagelj et al., 1992b: 70) or similarities used, some also define structural equivalence as: suppose that \( \equiv \) is an equivalence relation on \( U \) then \( \equiv \) is a structural equivalence if and only if for all \( a, b \in U, a \equiv b \) implies:

1. \( r_{bi} = r_{ai} \) for all \( i \in U \setminus \{a, b\} \) and
2. \( r_{ib} = r_{ia} \) for all \( i \in U \setminus \{a, b\} \).

In this definition, the loops and ties between the units that are checked for structural equivalence are simply ignored.

Generalizations or adaptations of other definitions (including those presented in Subsection 3.2.1) can be generalized in the same way. As already mentioned, the main difference between these definitions lies in the way they handle the loops and ties between the units that are checked for structural equivalence.

The definition by Breiger and Mohr (2004: 21-24) differs slightly since it is specially adapted for contingency tables. Their definition is essentially the same as that presented by Batagelj et
al. (1992b: 66), except that actual entries in the matrix $R$ are replaced by the interaction term in a log-linear model.

### 3.3.2 Regular equivalence

Borgatti and Everett (1992b: 102 – Definition 6) also provided a formal definition of regular equivalence for valued networks. Although in Borgatti and Everett (1992b) the definition was merely applied to networks measured on a nominal scale and not to those measured on an interval scale, the definition itself can also be used for networks where tie values are measured on an interval scale. Their definition may also be rewritten as follows (in a simplified form to make it more comparable with some other definitions presented below):

Suppose $\equiv$ is an equivalence relation on $U$ then $\equiv$ is a regular equivalence if and only if for all $a, b \in U, a \equiv b$ implies that:

1. for each $r_{ai}$, there exist such $r_{bj}$ that $r_{bj} = r_{ai}$, where $i \equiv j, i, j \in U$ and
2. for each $r_{ai}$, there exist such $r_{bj}$ that $r_{bj} = r_{ai}$, where $i \equiv j, i, j \in U$.

Alternative definitions of regular equivalence for valued networks measured on at least an interval scale can be formulated from:

1. two algorithms – REGGE (White, 1985a) and REGDI (White, 1985b) – for measuring the similarities and dissimilarities of units in terms of regular equivalence; and
2. ideas for defining block inconsistencies (the generalized blockmodeling approach) for valued networks presented by Batagelj and Ferligoj (2000: 12-13) that can also be used for regular equivalence.

Based on Batagelj and Ferligoj (2000: 12-13) the following definition of regular equivalence for valued networks can be formulated: suppose $\equiv$ is an equivalence relation on $U$ that induces (or corresponds to) partition $C$ then $\equiv$ is a regular equivalence if and only if for all $a, b \in U$ and all $X \in C$, $a \equiv b$ implies:

1. $\max_{i \in X}(r_{ai}) = \max_{i \in X}(r_{ia})$ and
2. $\max_{i \in X}(r_{ai}) = \max_{i \in X}(r_{ia})$.

This definition may also be written as follows (to make it more comparable with the one implied by REGGE that is presented later): suppose $\equiv$ is an equivalence relation on $U$ then $\equiv$ is a regular equivalence if and only if for all $a, b \in U, a \equiv b$ implies that for:

1. each $r_{ai}$, there exist such $r_{bj}$ that $r_{bj} \geq r_{ai}$, where $i \equiv j, i, j \in U$ and
2. each $r_{ai}$, there exist such $r_{bj}$ that $r_{bj} \geq r_{ai}$, where $i \equiv j, i, j \in U$.

Almost the same definition can be formulated from White’s (1985a) REGGE algorithm. His definition differs from the above one only in the case of directed networks since he believes that incoming and outgoing arcs should be treated jointly, not separately as in the approach of Batagelj and Ferligoj.
This can be written more formally as follows: suppose $\equiv$ is an equivalence relation on $U$ then $\equiv$ is a regular equivalence if and only if for all $a, b \in U$, $a \equiv b$ implies that for each pair $(r_{ai}, r_{ia})$, there exist such a pair $(r_{bj}, r_{jb})$ that:

1. $r_{bj} \geq r_{ai}$,
2. $r_{jb} \geq r_{ia}$ where
3. $i \equiv j$ and $i, j \in U$.

Based on White’s (1985b) second algorithm – REGDI – a slightly different definition can be formulated. In this algorithm all values, not only the maximal ones, must be equal. However, these equal values still do not have to appear on arcs to and from the same units. As is usual for regular equivalence it is enough that these units are equivalent.

This can be written more formally as follows: suppose $\equiv$ is an equivalence relation on $U$ then $\equiv$ is a regular equivalence if and only if for all $a, b \in U$, $a \equiv b$ implies that for each pair $(r_{ai}, r_{ia})$, there exist such a pair $(r_{bj}, r_{jb})$ that:

1. $r_{bj} = r_{ai}$,
2. $r_{jb} = r_{ia}$ where
3. $i \equiv j$ and $i, j \in U$.

All the presented definitions comply with all the requirements for a generalization of equivalence to valued networks. First, it is clear that the definitions include the definition for binary networks. Yet it should be noted these implied definitions for valued networks are an attempt to generalize two slightly different definitions (or interpretations of the definition) of regular equivalence.

It could be also said that the definitions capture the ‘essence’ of regular equivalence since they do not demand that two units must be connected to the same number of equivalent units. The definitions also comply with the requirement that the definition or at least the algorithm takes the continuous nature of the values into account.

The advantage of the definition implied by the approach of Batagelj and Ferligoj (2000: 12-13) is that it measures a tie between a unit and a cluster with a single number, which can easily be compared among different units.

3.3.3 $f$-regular equivalence

Another type of equivalence may be useful for valued networks. For now, let us call it $f$-regular (or function-regular) equivalence. This definition is not meant to be a strict generalization of regular equivalence. However, it tries to capture the idea that it is unnecessary for equivalent units to be equivalently connected to an individual unit, but only to a cluster of equivalent units. The idea of this set of possible equivalences is that a tie between a unit and a cluster of units can be adequately characterized by a function on the values of the ties connecting this unit to the units of the selected cluster. This idea can also be recognized in the definition developed in the previous section.
Two units are \( f \)-regular equivalent if the values of some function \( f \) over the values of arcs from (and separately to) each of these two units to a cluster of equivalent units are the same for both units and each cluster of equivalent units. If this function is the maximum we then get the definition of regular equivalence for valued networks implied by Batagelj and Ferligoj (2000: 12-13). However, other functions are also appropriate, especially the sum, the mean and the median. Especially in networks with negative tie values (in addition to positive), minimum would also be appropriate, while in networks with highly right skewed tie value distribution, geometric mean might be appropriate.

This definition is formally best presented in matrix terms and is given below. The definition is given for one-relation networks, however it can be generalized to multi-relational networks by demanding that the definition holds for all relations.

Suppose we have a network \( N = (U, R) \) and \( \equiv \) is an equivalence relation on \( U \) that induces (or corresponds to) clustering (or partition) \( C \) then \( \equiv \) is an \( f \)-regular equivalence (where \( f \) is a selected function, e.g. sum, maximum, mean etc.) if and only if for all \( a, b \in U \) and all \( X \in C \), \( a \equiv b \) implies:

1. \( f \left( \sum_{i \in X} r_{ia} \right) = f \left( \sum_{i \in X} r_{ib} \right) \) and
2. \( f \left( \sum_{i \in X} r_{ai} \right) = f \left( \sum_{i \in X} r_{bi} \right) \).

This could be considered another definition of regular equivalence for valued networks or a new type of equivalence. The definition implied by Batagelj and Ferligoj (2000: 12-13) could be named \( \text{max-regular} \) and this one \( \text{sum-regular}, \text{mean-regular} \) or more generally \( f \)-regular, where \( f \) stands for any suitable function. What a suitable function is will not be discussed here, although at least the sum and mean (in addition to the maximum) do qualify in at least some settings. The choice of function mainly depends on how we measure the strength of the tie between an individual and a cluster.

However, \( f \)-regular equivalences generally (\( \text{max-regular} \) does comply) do not comply with the requirements of good generalization. This does not pose a serious problem since the aim of these equivalences is not to generalize regular equivalence to valued networks but to present useful equivalences for valued networks that in some sense resemble regular equivalence. Some of them (namely the \( \text{sum-regular} \) equivalence) can, however, be adapted to fit these requirements, especially the first one, which is the most problematic for \( f \)-regular equivalences in general.

If we apply \( \text{sum-regular} \) equivalence to binary networks (where all line weights are equal to 1) we actually obtain the exact coloration presented by Everett and Borgatti (1994: 43). However, if we change the definition to the following one, set \( m \) to 1, and function \( f \) to the sum then the definition complies with regular equivalence.

\[ f \left( \sum_{i \in X} r_{ia} \right) = f \left( \sum_{i \in X} r_{ib} \right) \]

---

\[ 16 \text{ The relation } R \text{ is represented by matrix } R = [r_{ij}]_{n \times n}. \]
Suppose that we have a network $N = (U, R)^{17}$ and $\equiv$ is an equivalence relation on $U$ that induces (or corresponds to) clustering (or partition) $C$ then $\equiv$ is an \textit{f-regular equivalence at level} $m$ if and only if for all $a, b \in U$ and all $X \in C$, $a \equiv b$ implies both:

1. $f(r_{ia}) \geq m$ and $f(r_{ib}) \geq m$ or $f(r_{ai}) = f(r_{bi}) = 0$ and
2. $f(r_{ia}) \geq m$ and $f(r_{ib}) \geq m$ or $f(r_{ai}) = f(r_{bi}) = 0$.

The above definition is also an example of how some of the more standard definitions have to be modified so that one type of generalized blockmodeling, valued (generalized) blockmodeling, is sensitive to departures from these definitions. The notions type of generalized blockmodeling and valued blockmodeling is introduced in Chapter 6.

### 3.3.4 Other equivalences

Most other equivalences are usually not defined specifically for valued networks. However, some definitions are more easily adapted to valued networks. For example, a simple definition of automorphic equivalence is that automorphically equivalent units are indistinguishable in an unlabelled graph. Since two units that are connected to other units with ties of different strengths can be distinguished, they cannot be automorphically equivalent. Even here the problem arises if we wish to compute whether two units are automorphically equivalent and further if we want to compute some measure of deviation from automorphic equivalence. However, computational considerations are not addressed here.

Other definitions, especially those defined in terms of colorations and neighborhoods (for example, those in Everett and Borgatti, 1994), cannot be so simply interpreted for valued networks. Like in the Subsection 3.2.3, most other equivalences are not discussed here. In that section only exact coloration is described in more detail. Exact coloration is in fact a special case of the \textit{sum}-regular equivalence presented in the previous section. Therefore, we can say that \textit{sum}-regular equivalence is a possible generalization of exact coloration to valued networks. This generalization is especially suitable if the tie values actually represent weights that indicate multiple ties (to possible multiple identical units).

### 3.4 Generalized equivalence

The concept of generalized equivalence was introduced by Doreian et al.(1994). It is important to note that generalized equivalence is not a specific type of equivalence but more of a concept for building ‘custom’ equivalences by specifying allowed block types and, optionally, their positions. Generalized equivalence as a concept applies to both binary and valued networks provided that appropriate block types are defined. In this section only the basic concept is described. The topic is more exhaustively covered in Chapters 6 and 7.

---

17 The relation $R$ is represented by matrix $R = [r_{ij}]_{n \times n}$.
To better understand the concept two points should be revised. First, a relation $R$ is in generalized blockmodeling represented by a matrix $R = [r_{ij}]_{n \times n}$. Second, clustering $C$ also partitions relation $R$ into blocks. $R(C_i, C_j) = R \cap C_i \times C_j$.

Batagelj et al. (1992b) and Batagelj et al. (1992a) showed that structural and regular equivalence can be properly defined by a set of allowed ideal blocks and defined appropriate ideal blocks for these equivalences. They showed that structural equivalence corresponds to using complete and null ideal blocks, while regular equivalence corresponds to using regular and null ideal blocks (See Table 6.2 for definitions of these block types for binary networks through description of their ideal blocks).

They then proposed two generalizations of equivalence that lead to generalized equivalence and generalized blockmodeling. The first generalization is that blocks (produced by clustering based on generalized equivalence) can conform to different types of equivalence. The second generalization goes even further and states that it is not necessary that each block corresponds to equivalence; it is enough that it has a particular pattern where specific types of equivalences are special cases of more general patterns (Doreian et al., 1994: 2). They also presented a number of types of connections inside and between and within clusters, corresponding block types, and examples of patterns of ties within these block types (Doreian et al., 1994: 5-6).

### 3.5 Other criteria for blockmodeling

The desired properties of a partition are sometimes not explicitly specified by a definition of equivalence but only through a criterion function that measures the appropriateness of the partition. When used in blockmodeling, the criterion function is a function that evaluates a partition based on network data and returns a value. The best partition can be in principal that with the lowest or highest value of the criterion function, although it is desired that the criterion function can only take on non-negative values, that the partition with the lowest value of the criterion function is the best one, and that the criterion function has a zero value only when the partition completely matches the definition of the equivalence or other required properties.

Of course, a partition always defines equivalence and therefore also a criterion function that determines the partition, indirectly defining the equivalence.

Even in the previous sections and subsections in this chapter where different types of equivalences were discussed, the criterion function or a formula for computing similarities was sometimes used to not exactly define but to clarify the definition of the equivalence used or to explain how the ‘classical’ definition was either interpreted or generalized to valued networks.

However, sometimes the criteria for blockmodeling are not presented in terms of equivalence although, when a partition is being searched for, they could be. This is especially true with the
direct blockmodeling approach where a criterion function that matches some desired property of the clusters is searched for. Such examples that search for (cohesive) groups which match the definition of a group by Homans (1950: 84 in Freeman, 1993: 227) can be found in Freeman (1993) and Benisch (2004).

In some sense, the approach taken by Borgatti and Everett (1999) which aims to find the core/periphery structure can be placed in this group of approaches. On the other hand, it could be argued that it is more similar to the generalized blockmodeling approach (Doreian et al., 1994; 2004), especially in its pre-specified usage Batagelj et al., 1998).

3.6 Different approaches to blockmodeling

There are several different approaches to blockmodeling. Batagelj et al. (1992b: 66) proposed a division into indirect and direct approaches:

1. The indirect approaches firstly compute some measure of similarity or dissimilarity based on a selected measure of equivalence and then use one of the classical clustering techniques to uncover clusters of units.

2. Direct approaches, on the other hand, directly search for a partition that best fits the selected equivalence as is measured by a selected criterion function.

Both approaches are reviewed in their own chapters. Indirect approach is covered in Chapter 4 and the direct approach in Chapter 5. It should be noted that not all approaches can be easily classified in one of these two classes. One such approach is CONCOR (Breiger et al., 1975). Here it is described in the chapter on indirect approaches due to its lack of an explicit criterion function and that its first step is a computation of a correlation matrix (a measure of similarity), which is characteristic of indirect approaches. Especially if we include CONCOR in the indirect approach we can also say that this approach is historically speaking the first approach to blockmodeling.

The generalized blockmodeling approach is based on a direct approach. However, since this is the main topic of this dissertation it is covered separately in Chapter 7. The new approaches to the blockmodeling of valued networks are again covered separately in their own chapter, Chapter 7. The equivalences discussed in this chapter, especially those for valued networks, are one of the important foundations for many of the approaches presented in Chapters 4 to 7. The section on equivalences for valued networks, more precisely the subsection on regular and $f$-regular equivalences (Subsections 3.3.2 and 3.3.3), is where this chapter moves beyond a mere review of the literature.

All the approaches mentioned above are deterministic approaches, meaning that they are not based on some probabilistic model. Such an approach was developed by Snijders and Nowicki (1997; Nowicki and Snijders, 2001). However, this approach is only suitable for nominal valued networks.
As this dissertation focuses on the generalized blockmodeling of (interval) valued networks\(^{18}\), this approach is not reviewed. Only approaches that can analyze interval valued networks are considered. In spite of this, one approach that can only be used on binary networks is reviewed. This is binary generalized blockmodeling as this approach is the foundation on which the generalized blockmodeling of valued networks is developed.

All of these approaches are ‘global’ approaches. The approach is global and not local if ties, not in the ego-networks of the units (after the construction of any compound relations) evaluated for equivalence can influence the (degree) of equivalence of these two units. However, local approaches also exist such as those presented in Wu (1983), Everett et al. (1990), Winship and Mandel (1983) and elsewhere. Yet these approaches exceed the scope of this dissertation for the same reasons as some of the equivalence types mentioned earlier. They are unsuitable for blockmodeling in the sense of partitioning a matrix of a relation (or several matrices of relations) into blocks that exhibit some desired properties (patterns).

### 3.7 Conclusion

In this chapter an introduction to blockmodeling was given. Some definitions of this method were given and discussed. Blockmodeling is a method for finding clusters of units in networks and the ties between them. The two tasks can be done separately.

The review of the literature revealed there is an agreement that blockmodeling is a useful method for data reduction in networks. However, authors do not agree whether it can also be used to reveal roles and positions in social networks. The disagreement is also caused by different interpretations of the terms blockmodeling, role and position. Some additional notation is also introduced.

This is followed by three sections devoted to equivalences. Equivalences are a fundamental concept in social network analysis (Doreian, 1988a: 243). Section 3.2 reviews equivalences for binary networks. The two most widely used equivalences, structural and regular equivalence, are given special attention. It is noted that there is some ambiguity regarding regular equivalence. In Section 3.3 their extensions to valued networks are examined. While equivalences for binary networks are mostly well defined, there are some open issues regarding equivalences for valued networks. For regular equivalences, several open questions are added to the ambiguity already present in its version for binary networks. Due to this and the importance of equivalences for valued networks for my thesis, this section goes beyond a mere review of the literature. Regular equivalence is studied in detail and \(f\)-regular equivalence is introduced as useful equivalence for valued networks comparable to regular equivalence.

Generalized equivalence is presented in Section 3.4. Generalized equivalence is not a specific type of equivalence but more of a concept for building ‘custom’ equivalences. As it applies

\(^{18}\) Networks where tie values are measured on at least an interval scale.
both to binary and valued networks it is treated separately. It is closely related to generalized blockmodeling, the main topic of this thesis.

After that, other criteria (in addition to formal definitions of equivalences) for blockmodeling are reviewed. The chapter finishes with a review of the different approaches to blockmodeling. Distinctions among these approaches are noted. Those that either are able to analyze (interval\textsuperscript{19}) valued networks or are part of the generalized blockmodeling approach are selected to be covered in the following chapters.

\textsuperscript{19} In terms of the measurement scale.
4 INDIRECT APPROACHES TO BLOCKMODELING

As mentioned in the previous chapter, one of the broad approaches to blockmodeling is the indirect approach. This chapter is mainly devoted to a description of the general characteristics of this approach and of the measures of (dis)similarity that are usually used within this approach.

Batagelj et al. (1992b: 66) defined the indirect approach as the: ‘reduction to the standard data analysis problem (cluster analysis, multidimensional scaling) by determining a dissimilarity matrix between units which is compatible with the selected type of equivalence’. This procedure is described in Section 4.2. Although this is the main approach presented in this chapter, another approach that does not fully comply with this definition is also discussed here. This is the CONCOR (Breiger et al., 1975) approach. It is presented in the following section (Section 4.1).

At the beginning, blockmodeling was only associated with structural equivalence as it was the first equivalence used (for blockmodeling). At that time, the two algorithms most often used were STRUCTURE (Burt, 1976) and CONCOR (Breiger et al., 1975). The ideas of both algorithms are described in this chapter. CONCOR is described first in its separate section (Section 4.1) as it is not a classical indirect approach, at least not in the sense of the definition given by Batagelj et al. (1992b: 66). As STRUCTURE implemented a classical indirect approach to blockmodeling according to structural equivalence, the description of that approach described in 4.2 applies to it.

The sections following the section with a basic description of the indirect approach focus on individual equivalences or ways of computing (dis)similarities that are usually used in the procedure described in Section 4.2. The first of these is devoted to dissimilarities in terms of structural equivalence.

The next section, Section 4.4, is dedicated to (dis)similarities in terms of regular equivalence. In addition to reviewing existing algorithms for computing (dis)similarities in terms of regular equivalence, modified versions of these algorithms are developed. These modified versions are aimed at either making the (dis)similarities (more) compatible with different versions of regular equivalence (discussed in Section 3.3.2) or at adapting these (dis)similarities to special kinds of data. The last similarity reviewed in Section 4.4.4 is structural similarity introduced by Brandes and Lerner (2004). Structural similarity can represent the similarity of units in the network based on several structural characteristics of the network.

At the end, a short conclusion emphasizes parts of the chapter that go beyond a simple review of the literature.
4.1 CONCOR

As mentioned before, CONCOR (Breiger et al., 1975) is not a typical indirect approach as it does not first compute a similarity or dissimilarity measure but directly produces a divisive hierarchical clustering. However, at least the first part of this statement is not completely true. As already noted by Sailer (1978: 77), the first step of CONCOR is the computation of a correlation matrix, a measure of similarity, as is common to all indirect approaches. But here the CONCOR's similarity with the indirect approaches ends since it does not treat this correlation matrix as an input to a standard data analysis problem (cluster analysis, multidimensional scaling). CONCOR stands for the ‘CONvergence of iterated CORrelations’, which is a suitable description of what it does. CONCOR repeatedly computes correlations between the rows (or columns) of a matrix. In the first iteration this matrix is a matrix of the relation(s) analyzed, while in all other iterations it is a matrix of correlations computed in a previous iteration. This process converges when all elements of the matrix are either 1 or -1. These 1's and -1's appear in such a pattern that the units analyzed can be partitioned into two clusters so that all the 1's are in the diagonal blocks and all the -1's are in the off-diagonal blocks. CONCOR can then be repeated for sub-matrices defined by the partition obtained in a previous step and thus produce a series of ever finer partitions. Therefore, CONCOR can be thought of as a divisive hierarchical clustering method (Wasserman and Faust, 1994: 376-378).

4.2 Basic description of the indirect approach

Here the approach that fully complies with the definition by Batagelj et al. (1992b: 66) is described. Although there are many possibilities in using this approach we can still only talk about one approach. The approach was probably first implemented in STRUCTURE (Burt, 1976).

The approach involves two stages. First, a similarity (which might have to be converted to dissimilarity) or dissimilarity compatible with the selected equivalence has to be computed. This dissimilarity\(^{20}\) is then used as an input for standard data analysis techniques for finding clusters, usually hierarchical clustering. This approach can be used for any equivalence (not only structural) if a compatible similarity or dissimilarity measure exists.

Based on Batagelj et al. (1992b: 72) we can say that a dissimilarity \(d\) is compatible with an equivalence \(\equiv\), if and only if for any two units \(a, b \in U\) \(a \equiv b\) implies \(d(a, b) = 0\). Suitable dissimilarities are discussed in the following sections separately for different equivalences.

However, the new step does not depend on the equivalence or dissimilarity used. Computed dissimilarities are used as an input for some standard data analysis techniques for finding

\(^{20}\) Or similarity. Only the term dissimilarity is used since dissimilarities are usually used as an input to classical data analysis techniques. Even if similarities are computed, they are usually converted to dissimilarities. However, in general, similarities can also be used, assuming that appropriate methods exist.
clusters. Although in principle any method can be employed that uses as an input dissimilarities and produces a partition, hierarchical clustering is usually used.

Pajek 1.05 (Batagelj and Mrvar, 2005a, 2005b) provides the greatest options of the reviewed programs since several hierarchical clustering methods can be used:

1. **General** – the Lance-Williams recurrence formula is used that ‘gives the distance between a cluster k and a cluster (ij) formed by the fusion of the clusters (I and j) as\[ d(C_i \cup C_j, C_k) = \alpha_i d(C_k, C_i) + \alpha_j d(C_k, C_j) + \beta d(C_i, C_j) + \gamma|d(C_k, C_i) - d(C_k, C_j)| \]
   (Ferligoj, 1989: 71)\(^{21}\), where \(d(X,Y)\) represents the distance between clusters X and Y and \(\alpha_i\), \(\alpha_j\), \(\beta\) and \(\gamma\) are parameters that can be set. Many other standard hierarchical techniques can be contained within this formula by a suitable choice of parameters \(\alpha_i\), \(\alpha_j\), \(\beta\) and \(\gamma\), for example minimum or single linkage or nearest neighbor, maximum or complete linkage or furthest neighbor, average linkage, centroid linkage median linkage and Ward's method (Ferligoj, 1989: 72)\(^{22}\).

2. **Minimum or single linkage or nearest neighbor** – the minimum distance between two clusters is used or, more precisely, the minimum of distances between pairs of objects where one object is in one cluster and the other is in another, is used.

3. **Maximum or complete linkage or furthest neighbor** – the maximum distance between two clusters is used or, more precisely, the maximum of distances between pairs of objects where one object is in one cluster and the other is in another, is used.

4. **Average linkage** – the average distance between two clusters is used or, more precisely, the average of distances between pairs of objects where one object is in one cluster and the other is in another, is used.

5. **Ward** – I assume the following formula (which is based on the Lance-Williams recurrence formula (Ferligoj, 1989: 70:72)\(^{23}\)):
   \[
d(C_i \cup C_j, C_k) = \frac{(n_i + n_k)}{(n_i + n_j + n_k)} d(C_k, C_i) + \frac{(n_j + n_k)}{(n_i + n_j + n_k)} d(C_k, C_j) - \frac{n_k}{(n_i + n_j + n_k)} d(C_i, C_j)
   \]
   In order to get Ward's (Ward, 1963) original procedure, the squared Euclidean distance should be used as a measure of similarity. For use on network data, the squared Euclidean distance must be corrected, as presented in the next section.

6. **Squared Ward** – uses the same formula above, however, the distances are squared before they are used. This allows the direct use of Euclidean distance for the correct Ward's method.

---

\(^{21}\) See Everitt et al. (2001: 61) for an English source.

\(^{22}\) See Everitt et al. (2001: 61-63) for an English source.

\(^{23}\) See Everitt et al. (2001: 61, 63) for an English source.
STRUCTURE 4.2 (Burt, 1991: 136) provides two options, namely Ward's method and single linkage. Ucinet 5 (Borgatti et al., 1999) only used single link hierarchical clustering for both structural and regular equivalence, while Ucinet 6 (Borgatti et al., 2002) allows the user to choose between single, complete and average linkage. All programs also allow the user to export the obtained dissimilarity matrix for use in other programs.

4.3 Structural equivalence

Several dissimilarities and similarities are used to capture structural equivalence. Use of the measure also depends on the definition of the equivalence employed since several definitions were used, as described in Subsection 3.2.1 and briefly in 3.3.1.

As already pointed out in Subsections 3.2.1 and 3.3.1 (sections on structural equivalences for binary and valued networks), the main differences among the various definitions and, as a consequence, the dissimilarities or similarities lies in the way they handle the loops and ties between units for which structural equivalence is evaluated. The other difference among different dissimilarities and similarities is the type of dissimilarity or similarity used, such as Euclidean dissimilarity, Manhattan dissimilarity or more generally any Minkowski dissimilarity or some other measure such as correlation or covariance.

A number of dissimilarity and similarity measures have also been designed especially for binary data, for example those reviewed in Batagelj and Bren (1995). Others have been designed for network data based on the similarity or overlap of neighborhoods, for example those implemented in Pajek (Batagelj and Mrvar, 2005b: 30).

The appropriate form of dissimilarity for each option of the definitions described in the Subsections 3.2.1 and 3.3.1 is now presented in the case of Euclidean dissimilarity:

1. The loops and ties between the units that are checked for structural equivalence are not treated any differently as any other cells in the matrix (as in the definitions by Lorain and White (1971: 81) and Burt (1976: 96)):

   Euclidean distance: \( d_E(X_i, X_j) = \sqrt{\sum_{k=1}^{n} ((r_{ik} - r_{jk})^2 + (r_{ki} - r_{kj})^2)} \)

2. The loops and ties between the units that are checked for structural equivalence are simply ignored (as implemented in the ‘ignore’ method for handling diagonal values in Ucinet 5 (Borgatti et al., 1999)):

   Truncated Euclidean distance (Faust, 1988 in Batagelj et al., 1992: 70):

   \[ d_S(X_i, X_j) = \sqrt{\sum_{k=i,j}^{n} ((r_{ik} - r_{jk})^2 + (r_{ki} - r_{kj})^2)} \]
3. The special role/status of the loops and ties between the units that are checked for structural equivalence is recognized (as in the definition by Batagelj et al., 1992b: 66):

Corrected Euclidean-like dissimilarity (Burt and Minor, 1983 in Batagelj et al., 1992: 71):

\[ d_{c}(p)(X_i, X_j) = \sqrt{\sum_{k=1}^{n} \left( (r_{ik} - r_{jk})^2 + (r_{li} - r_{lj})^2 + p((r_{ij} - r_{ji})^2 + (r_{ji} - r_{ij})^2) \right)} \]

In order for this dissimilarity to fully comply with the definition by Batagelj et al. (1992b: 66), p must be greater than one, usually 1 or 2. If p = 1, we get:

\[ d_{c}(p=1)(X_i, X_j) = \sqrt{\sum_{k=1}^{n} \left( (r_{ik} - r_{jk})^2 + (r_{li} - r_{lj})^2 + ((r_{ij} - r_{ji})^2 + (r_{ji} - r_{ij})^2) \right)} \]

The last dissimilarity (Corrected Euclidean-like dissimilarity) also includes Truncated Euclidean distance as a special case when p = 0.

The principles followed in these formulas for Euclidean(-like) dissimilarities can also be applied to other types of dissimilarities of similarities (for example, Manhattan dissimilarity or more generally any Minkowski dissimilarity, some type of correlation, or covariance).

Batagelj et al. (1992b: 70-73) reviewed several dissimilarities for structural equivalence based on either Euclidean or Manhattan distance and showed that not all are compatible with structural equivalence. In fact, they showed that of these dissimilarities only those corrected in a similar way as Corrected Euclidean-like dissimilarity (Burt and Minor, 1983 in Batagelj et al., 1992b:71) are compatible with structural equivalence. It should be noted that in this dissertation the same definition of structural equivalence as presented in Batagelj et al. (1992b: 65-66) is used. Yet other authors may use slightly different definitions of structural equivalence (as shown in Section 3.2.1). Other dissimilarities (those presented above under 1. and 2.) are compatible with these definitions.

4.4 Regular equivalence

There is no ‘closed form’ measure of similarity or dissimilarity compatible with regular equivalence. However, there is an algorithm that produces in iterations a sequence of measures of similarity or dissimilarity (depending on the version) in terms of regular equivalence (White, 2005). For blockmodeling purposes, this (dis)similarity matrix must be analyzed using an appropriate clustering algorithm. Hierarchical clustering algorithms have usually been used for this purpose. For example, Ucinet 5 (Borgatti et al., 1999) uses single linkage hierarchical clustering to find an appropriate partition. However, based on my

---

24 No measure exists that can be computed in closed form.
experience, other clustering algorithms such as Ward's method or complete linkage perform better.

Regular equivalence for valued networks was first implemented in the REGGE (a version of REGE\textsuperscript{25}) algorithm (White, 2005; 1985a). Several versions of REGE exist and some are presented in this thesis in more detail. The discussion is limited to versions of REGE designed for valued networks that are measured at least on an interval scale. All versions have a basic principle in common, which can be seen in the FORTAN program written by White (1985a) to implement the regular equivalence introduced by White and Reitz (1983).

However, as already noted by Borgatti and Everett (1993:385) ‘…there are certain aspects of REGE’s operation that are not altogether clear’.

A few descriptions of the algorithm can be found in the literature, however they are not sufficient. A key paper by White and Reitz (1985) remains unpublished\textsuperscript{26}. In addition, it does not describe the algorithm (White, 1985a) that can be found on Douglas R. White’s\textsuperscript{27} web page (White, 2005)\textsuperscript{28}, while the description given by Borgatti and Everett (1993) focuses on binary networks and does not indicate how REGE handles the tie values. Some additional papers have been written (White,1984a; White, 1980; 1982; and 1984b in Borgatti and Everett, 1993) on this topic. Unfortunately, they also remain unpublished.

In this section the algorithm and some of its variants\textsuperscript{29} are described, initially in the short description below. However, much more can be learned by means of the formulas used by the algorithms to compute similarities or dissimilarities in terms of regular equivalence. Hopefully, the formulas and descriptions that accompany them are sufficient to allow an understanding of the algorithm and differences between the several variants that exist.

REGE evaluates (the degree of) equivalence of units $a$ and $b$ by trying to match every link\textsuperscript{30} (taking values into account) of unit $a$ by the most similar link of unit $b$ to most/an equivalent unit and vice versa (every link of unit $b$ to most/an equivalent link of unit $a$).

If the best match (link) does not have identical tie values (there are two values per link) or if the other unit in the link of unit $a$ is not completely equivalent to the other unit in the link of unit $b$, the error depends on the difference of the tie values compared relative to their

\textsuperscript{25} In this text, the term REGE is used for all versions of the REGE algorithm.

\textsuperscript{26} To the best of my knowledge, the first published paper containing the formulas for computing REGE similarities was written by Faust (1988, 325-328).

\textsuperscript{27} The author of the REGE algorithm and coauthor of the original definition of regular equivalence.

\textsuperscript{28} Based on the comparison of returned results, I assume that the same algorithm is used in Ucinet 5 (Borgatti et al., 1999).

\textsuperscript{29} Only the variants aimed at regular equivalence and data measured on at least an interval scale are considered here.

\textsuperscript{30} It should be noted that in REGE developed by White (1985a, 1985b – both versions) a link represents a pair of arcs – an incoming and an outgoing arc, if both exist. Therefore, a vector of length 2 represents a link in a valued network. Using the notation introduced in Section 2.2, the link between units $a$ and $i$ is a vector $(r_{ai}, r_{ia})$. 
magnitude and on the similarity (in terms of regular equivalence) of the other units in the links compared. Regular equivalence does not demand that two units which are equivalent are connected to the same number of equivalent units. REGE captures this feature by allowing that different links of a can be matched by the same link of b. It could be said that REGE algorithms implicitly define regular equivalence for valued networks.

Yet the two versions – REGGE (White, 1985a) and REGDI (White, 1985b) – implicitly provide two different definitions of regular equivalence. This was discussed in Subsection 3.3.2.

It should be noted that REGE uses a slightly different interpretation of the definition of regular equivalence than is usually used in the generalized blockmodeling approach (and some other approaches) and in this text. Nevertheless, the logic of the algorithm can be used and even the algorithm itself can be easily adapted to the interpretation used in generalized blockmodeling. This was already noted by Borgatti and Everett (1993). This adaptation is presented in Subsection 4.4.2.

4.4.1 Existing versions (or descriptions) of REGE

REGE’s notations

- $i_j M_{km}$ … Matches of the ties between unit i and unit k to the ties between unit j and unit k
- $i_j \text{Max}_{km}$ … Maximum possible matches of the ties between unit i and unit k to the ties between unit j and unit k
- $t$ … iteration (number)
- $E_{ij}^{t}$ … Degree of equivalence between unit i and unit j in iteration t.

$SB = [sb_{ij}]_{n \times n}$ … A symmetric binary relation based on R represented by binary matrix SB with cells $sb_{ij}$, where $sb_{ij} = 1$, if $r_{ij} + r_{ji} > 0$ and 0 otherwise.

$B = [b_{ij}]_{n \times n}$ … A binary relation based on R represented by binary matrix B with cells $b_{ij}$, where $b_{ij} = 1$, if $r_{ij} > 0$ and 0 otherwise.

* $m \text{ max}$ indicated that in the denominator, an element that corresponds to the maximum in the numerator is selected. The optional indices in brackets indicate to which of the maximums in the numerator the element must correspond.

Several31 early versions of the REGE algorithm are described in an unpublished paper by White and Reitz (1985, 14-15,17-18). However, out of those, here only that which can also be

---

31 White and Reitz (1985, 14-20) presented two main versions of the algorithm presented below that differ only in the case of multi-relational networks. Since here only one-relational networks are addressed, this is not relevant. Also, they present two starting options for an initial similarity matrix ($E^0$), of which only one is presented here. In addition, they present a similar set of algorithms which compute regular distances instead of regular similarities. They are also not covered here (as they were never officially published), however a newer version is discussed (see the REGDI algorithm below).
found in a published paper written by Faust (1988, 325-328) is presented since the others were never officially published.

**REGGE-O(Old): REGE algorithm from White and Reitz (1985, 18)**

\[
\begin{align*}
M_{km}^{ij} &= \min\left(r_{ik}, r_{jm}\right) + \min\left(r_{ki}, r_{mj}\right) \\
\text{Max}_{km}^{ij} &= \max\left(r_{ik}, r_{jm}\right) + \max\left(r_{ki}, r_{mj}\right) \\
E_{ij}^{t+1} &= \frac{\sum_{k=1}^{n} \max_{m=1}^{n} \left(E_{km}^{t} \left(j_{M_{km}^{ij}} + j_{\text{Max}_{km}^{ij}}\right)\right)}{\sum_{k=1}^{n} \max_{m=1}^{n} \left(j_{\text{Max}_{km}^{ij}} + j_{\text{Max}_{km}^{ij}}\right)}
\end{align*}
\]

The first two equations have to be computed only once since they do not include any reference to \(t\), the iteration number. The third equation presented the computation of similarity matrix \((E_{ij}^{t+1})\) in iteration \(t + 1\) based on the similarity matrix \((E_{ij}^{t})\) in the previous iteration and the quantities computed in the above formulas. For the first iteration, a matrix filed with 1s is used as a similarity matrix from the previous iteration \((E^{0})\). The formulas, however, have a strange feature. As we can see, in the first two formulas the quantities \(ijM_{km}\) and \(ij\text{Max}_{km}\) have no reference to the iteration number \((t)\). However, in the third formula it has an index \(t\), which indicates the iteration number. This is probably left over from an earlier version of the algorithm, where \(ijM_{km}\) did depend on the iteration (White, 1984a).

Another inconsistency with both the definition of regular equivalence (White and Reitz, 1983: 200) and the description of REGE (plus examples) in Borgatti and Everett (1993: 364-369) can be found. First, in the numerator the maximum should be computed separately for \(ijM_{km}\) and separately for \(jiM_{km}\), however in this formula it is computed for \(ijM_{km} + jiM_{km}\). Similarly, the corresponding \(ijM_{km}\) should be chosen in the denominator. The denominator is also computed wrongly since \(ij\text{Max}_{km}\) should only be used in the sum if \(r_{ik} + r_{ki}\) is greater than 0 (only matches for links are searched for, while 0 indicates an absence of a link)\(^{32}\). However, one detail still has to be established. This is which \(ij\text{Max}_{km}\) is to be chosen where there are several maximal \(ijM_{km}\). In that case, we select the minimal \(ij\text{Max}_{km}\) out of those that correspond to the maximal \(ijM_{km}\). In this way, if an exact match to \(r_{ik}\) exists, it will be selected.

The notation was also corrected to emphasize this and the ambiguous notation \(\max_{m}^{*}\) was replaced by more conventional mathematical operators. If we correct all these inconsistencies with the definition and ambiguities, we obtain the formulas written below.

\(^{32}\) The option where \(ij\text{Max}_{km}\) would be used in the sum in the denominator even when \(r_{ik} + r_{ki}\) is 0 is also possible. However, it does not match the definition of regular equivalence (White and Reitz, 1983: 200) and the description of REGE (plus examples) in Borgatti and Everett (1993: 364-369).
REGGE-OC (Old Corrected): Corrected REGGE-O algorithm

\[ M_{km} = \min (r_{ik} \cdot r_{jm}) + \min (r_{ki} \cdot r_{mj}) \]
\[ \text{Max}_{km} = \max (r_{ik} \cdot r_{jm}) + \max (r_{ki} \cdot r_{mj}) \]

\[ E^{t+1}_{ij} = \frac{\sum_{k=1}^{n} \left( \max (E^t_{kmij} M_{km}) + \max (E^t_{kmji} M_{km}) \right) - \sum_{k=1}^{n} (r_{ik} + r_{ki} + r_{jk} + r_{kj})}{\sum_{k=1}^{n} \left( \min_{\text{arg max}} \left( E^t_{kmij} M_{km} \right) \right)} \]

It may be noticed that the formulas for \( ijM_{km} \) and \( ij\text{Max}_{km} \) do not change. The recursive formula changes in a way that \( ijM_{km} \) and \( jiM_{km} \) are now maximized separately (over \( m \) for a given \( i, j, \) and \( k \)) and the corresponding \( ij\text{Max}_{km} \) is similarly chosen. This version produces the same exact regular equivalence classes as the algorithm REGDI which is presented later.

However, this is still not the formula used in REGGE (White, 1985a). That formula uses a simpler denominator and is presented in the formulas below. The simplification of the denominator means that an arc with any value greater or equal to the value of the arc for which a match is searched perfectly matches this arc. Yet the vice versa situation is not true. This was not the case in the above algorithms where for a perfect match the arcs had to have identical values. The same is also true for REGDI, an algorithm presented later. This characteristic can also be found in CATREGE (Borgatti and Everett, 1993), however CATREGE assumes that values are measured on a nominal scale.

REGGE: REGE algorithm as implemented in REGGE (White, 1985a)\(^{33} \):

\[ M_{km} = \min (r_{ik} \cdot r_{jm}) + \min (r_{ki} \cdot r_{mj}) \]
\[ E^{t+1}_{ij} = \frac{\sum_{k=1}^{n} \left( \max (E^t_{kmij} M_{km}) + \max (E^t_{kmji} M_{km}) \right) - \sum_{k=1}^{n} (r_{ik} + r_{ki} + r_{jk} + r_{kj})}{\sum_{k=1}^{n} \left( \min_{\text{arg max}} \left( E^t_{kmij} M_{km} \right) \right)} \]

White (1985b) also developed a distance version of REGE, which he called REGDI.

REGDI: REGE algorithm as implemented in REGDI (White, 1985b):

\[ M_{km} = (r_{ik} - r_{jm})^2 + (r_{ki} - r_{mj})^2 \]
\[ E^{t+1}_{ij} = \min \left\{ \sum_{k=1}^{n} \left( \min_{\text{arg max}} \left( E^t_{kmij} M_{km} , E^t_{kmji} \left( r_{ik}^2 + r_{ki}^2 \right) \right) + \min_{\text{arg max}} \left( E^t_{kmji} M_{km} , E^t_{kmij} \left( r_{jk}^2 + r_{kj}^2 \right) \right) \right) \right\} \]

\[ \sum_{k=1}^{n} (r_{ik}^2 + r_{ki}^2 + r_{jk}^2 + r_{kj}^2) \]

\(^{33} \) The formula was confirmed by the author of the original REGE algorithm, Douglas R. White, through personal communication. Professor White also helped the author of this thesis gain a better understanding of the REGGE algorithm.
In the formulas for REGDI, $E^t$ is a dissimilarity matrix for the $t$-th iteration. The computations here are not as clear as in the previous cases. If we look closely at the formulas, we see that in the evaluation of each pair of links either only equivalence (through dissimilarity in terms of regular equivalence) of the units at the other end (not those currently evaluated for regular equivalence) of the links or only the difference between the tie values comprising the two links is used in the contribution of the ‘base’ link to the dissimilarity in terms of regular equivalence. The ‘base’ link is the link between units $i$ and $k$, that is a vector $(r_{ik}, r_{ki})$.

However, the numerator cannot simply be

$$
\sum_{k=1}^{n} \left( \min_{m=1}^{n} (M_{km} E^t_{km}) + \min_{m=1}^{n} (M_{km} E^t_{km}) \right)
$$

since this quantity $\min_{m=1}^{n} (M_{km} E^t_{km})$ would always be 0 since $E^t_{km}$ for $k = m$ is always 0. The algorithm used is an attempt to solve this problem. It forces the algorithm to focus on differences when the units at the other end of links are quite similar (in the sense of regular equivalence) and to focus on the dissimilarities in the sense of regular equivalence when the differences in the tie values are relatively small.

### 4.4.2 Suggested modifications of REGE

As already mentioned, REGE does not fully comply with the interpretation of the definition of regular equivalence used in generalized blockmodeling. Therefore, I have developed a few modified versions of REGE. One of them, REGGE-OW, fully complies with the interpretation of the definition used in generalized blockmodeling, while the rest represent only a step in this direction, namely they incorporate the philosophy whereby each arc in a dyad is treated separately.

The definition used in generalized blockmodeling rests on the proposition of Batagelj et al. (1992a: 125) that if $C = \{C_i\}$ is a partition corresponding to regular equivalence, then $R(C_u, C_v)$ is either null (empty) or has a property whereby there exists at least one 1 (tie) in each of its rows and in each of its columns. Borgatti and Everett (1992b: 101-107) presented similar ideas. Although this definition is suitable for binary networks, the logic that arcs $r_{ij}$ and $r_{ji}$ should be treated separately can easily be applied to valued networks.

Only a slight modification is needed to modify the REGE algorithm(s) to comply with this definition. The modified algorithms are presented below. They are named ‘One-Way’ (OW) since here each (arc) way of the dyad is evaluated separately.
REGGE-OCOW (One-Way): One-way version of the REGGE-OC algorithm

\[
E_{ij}^{t+1} = \frac{\sum_{k=1}^{n} \left( \max_{m=1}^{n} \left( E_{km}^t \min(r_{ik}, r_{jm}) \right) + \max_{m=1}^{n} \left( E_{km}^t \min(r_{ki}, r_{mj}) \right) \right) + \max_{m=1}^{n} \left( E_{km}^t \min(r_{kk}, r_{im}) \right) + \max_{m=1}^{n} \left( E_{km}^t \min(r_{kj}, r_{mi}) \right) }{\sum_{k=1}^{n} (r_{ik} + r_{ki} + r_{jk} + r_{kj})}
\]

The only modification used here is that we no longer search for the maximum of \( M_{km} = \min(r_{ik}, r_{jm}) + \min(r_{ki}, r_{mj}) \), but separately for the maximum of \( \min(r_{ik}, r_{jm}) \) and separately of \( \min(r_{ki}, r_{mj}) \). Similarly, suitable maximums are chosen separately.

REGGE-OV (One-Way): One-way version of the REGGE algorithm

\[
E_{ij}^{t+1} = \frac{\sum_{k=1}^{n} \left( \max_{m=1}^{n} \left( E_{km}^t \min(r_{ik}, r_{jm}) \right) + \max_{m=1}^{n} \left( E_{km}^t \min(r_{ki}, r_{mj}) \right) \right) + \max_{m=1}^{n} \left( E_{km}^t \min(r_{kk}, r_{im}) \right) + \max_{m=1}^{n} \left( E_{km}^t \min(r_{kj}, r_{mi}) \right) }{\sum_{k=1}^{n} (r_{ik} + r_{ki} + r_{jk} + r_{kj})}
\]

In this version of REGE (REGGE-OV) the definition of regular equivalence used for valued networks is actually the same as implied in the implicit approach of Batagelj and Ferligoj (2000: 12-13), as presented in Subsection 3.3.2 and named max-regular equivalence in Subsection 3.3.3.

REGDI-OV (One-Way): One-way version of the REGDI algorithm

\[
E_{ij}^{t+1} = \min \left\{ \frac{\sum_{k=1}^{n} \left( \min_{m=1}^{n} \left( r_{ik}^2 + r_{ki}^2 \right) + \min_{m=1}^{n} \left( r_{kk}^2 + r_{im}^2 \right) \right) + \min_{m=1}^{n} \left( r_{kj}^2 + r_{mi}^2 \right) }{\sum_{k=1}^{n} (r_{ik}^2 + r_{ki}^2 + r_{jk}^2 + r_{kj}^2)} \right\}
\]

4.4.3 REGE for valued networks where the tie values are influenced by the ‘size’ of the units

When analyzing valued networks, we often encounter the problem of the appropriate scaling of tie values. Typical examples where this problem occurs are trade networks between countries or other entities, where we want to cluster units based on the relative strength (or
pattern) and not on the absolute strength of ties. The same tie can be insignificant for one unit (a large country, as it presents for example only a small fraction of its total imports) and very important for some other unit (a small county, since it can account for 50% or more of its total exports).

If we use just row or just column normalization (dividing the values in the matrix by the appropriate row/column sum), we are analyzing the network merely from the one ‘way’ view. In the case of the trade network example, we are analyzing the network only from the import or export view. Even if we undertake both row and column normalization (usually by dividing the matrix by the square roots of the appropriate row and column sums), we are not taking both views into account but rather some compromise between them. A similar compromise is to iteratively use row and column normalization until convergence as, for example, used by Doreian (1988b), which additionally does usually not converge in situations where some rows or columns only have one non-zero cell.

A solution to this problem was proposed by Nordlund (2007, 61). He proposes that both row and column normalization should be done (separately). In this way, two matrices should be obtained (one row-normalized and one column-normalized). Then the row-normalized matrix should be checked for row-regular blocks and the column-normalized matrix for column-regular blocks\(^{34}\). A block is then declared regular if the conditions for these two block types (one in each matrix) are fulfilled to the predefined extent\(^{35}\). However, the approach described by Nordlund (2007, 61-63) can be improved in two ways:

1. Nordlund assumes that REGE does a reasonable job of identifying regular equivalence classes. I find this assumption highly questionable. If absolute tie values are not appropriate for determining whether a certain block is regular (with which I agree), then I doubt that they are appropriate for finding regularly equivalent classes.

2. Nordlund transforms the row- and column-normalized matrices into binary matrices by using a global cut-off value on them. I argue that it would be better to use the original (valued) normalized matrices for assessing how well certain blocks correspond to row- and column-regular block types (in the normalized matrices) and consequently to the regular block type.

The first point is addressed below by modifying the versions of the REGE algorithm suggested above. The second point is addressed in Section 6 – Generalized blockmodeling.

As the modified algorithm will work with two matrices these two matrices have to be defined.

\(rR\) … row-normalized matrix \(rR = R \times 1\), with elements \(rr_{ij}\).

\(cR\) … column-normalized matrix \(cR = 1 \times R\), with elements \(cr_{ij}\).

\(^{34}\) See Section 6 – Generalized blockmodeling for details about row- and column-regular blocks.

\(^{35}\) See Nordlund (2007, 3-5) for details.
REGGE-NM (Normalized Matrices): REGGE algorithm modified to use row- and column-normalized matrices

\[ \sum_{k=1}^{n} \left( \max_{m=1}^{n} \left( E_{km}^{t} \min \left( r_{ik}, r_{jm} \right) \right) + \max_{m=1}^{n} \left( E_{km}^{t} \min \left( cr_{ki}, cr_{mj} \right) \right) \right) \]

\[ E_{ij}^{t+1} = \sum_{k=1}^{n} \left( r_{ik} + cr_{ki} + r_{jk} + cr_{kj} \right) \]

REGGE-OWNM (Normalized Matrices): REGGE-OW algorithm modified to use row- and column-normalized matrices

\[ \sum_{k=1}^{n} \left( \max_{m=1}^{n} \left( E_{km}^{t} \min \left( r_{ik}, r_{jm} \right) \right) + \max_{m=1}^{n} \left( E_{km}^{t} \min \left( cr_{ki}, cr_{mj} \right) \right) \right) \]

\[ E_{ij}^{t+1} = \sum_{k=1}^{n} \left( r_{ik} + cr_{ki} + r_{jk} + cr_{kj} \right) \]

REGGE-OCOWNM (Normalized Matrices): REGGE-OCOW algorithm modified to use row- and column-normalized matrices

\[ \sum_{k=1}^{n} \left( \max_{m=1}^{n} \left( E_{km}^{t} \min \left( r_{ik}, r_{jm} \right) \right) + \max_{m=1}^{n} \left( E_{km}^{t} \min \left( cr_{ki}, cr_{mj} \right) \right) \right) \]

\[ E_{ij}^{t+1} = \sum_{k=1}^{n} \left( r_{ik} + cr_{ki} + r_{jk} + cr_{kj} \right) \]

REGGE-OCNM (Normalized Matrices): REGGE-OC algorithm modified to use row- and column-normalized matrices

\[ \sum_{k=1}^{n} \left( \max_{m=1}^{n} \left( E_{km}^{t} \min \left( r_{ik}, r_{jm} \right) \right) + \max_{m=1}^{n} \left( E_{km}^{t} \min \left( cr_{ki}, cr_{mj} \right) \right) \right) \]

\[ E_{ij}^{t+1} = \sum_{k=1}^{n} \left( r_{ik} + cr_{ki} + r_{jk} + cr_{kj} \right) \]
REGDI-OWNM (Normalized Matrices): REGDI-OW algorithm modified to use row- and column-normalized matrices

\[
E^t_{ij} = \min \left( \sum_{k=1}^{n} \left( \min_{m=1}^{n} \left( \max \left( \left( r_{ik} - r_{jm} \right)^2, E^t_{km} r_{ik}^2 \right) \right) + \min_{m=1}^{n} \left( \max \left( \left( c_{rk} - c_{rm} \right)^2, E^t_{km} c_{rk}^2 \right) \right) \right), 1 \right)
\]

REGDI-NM (Normalized Matrices): REGDI algorithm modified to use row- and column-normalized matrices

\[
M_{km} = \left( r_{ik} - r_{jm} \right)^2 + \left( c_{rk} - c_{rm} \right)^2
\]

\[
E^t_{ij} = \min \left( \sum_{k=1}^{n} \left( \min_{m=1}^{n} \left( \max \left( M_{km}, E^t_{km} \left( r_{ik}^2 + c_{rk}^2 \right) \right) \right) + \min_{m=1}^{n} \left( \max \left( M_{km}, E^t_{km} \left( r_{ik}^2 + c_{rk}^2 \right) \right) \right) \right), 1 \right)
\]

4.4.4 Comparison of versions of REGE

A relatively large number of versions of REGE were described in this section. As pointed out when existing approaches were presented, REGGE and REGDI comply with two different definitions of regular equivalence, as was discussed in Subsection 3.3.2. The definition used in REGGE is less stringent since it only requires that the maximal tie values that the two (units are evaluated by pairs) units evaluated for equivalence have with units of a certain equivalence class (cluster) are the same. The definition used by REGDI, on the other hand, requires that all unique tie values of one of these (those two evaluated for equivalence) units have to a given equivalence class (cluster) are matched by tie values that the other of the two units evaluated have to the same equivalence class (cluster). REGGE-OC uses the same equivalence as REGDI. All these versions of REGE evaluated two possible ties between two units (one in each direction) jointly. This is not appropriate for generalized blockmodeling as it would mean that blocks \( R(X,Y) \) and \( R(Y,X) \) would have to be evaluated jointly. Definitions of regular equivalence that do not require this joint treatment were also presented in Subsection 3.3.2.

The previously mentioned versions of REGE modified to comply with these definitions are marked with the original name followed by ‘-OW’ (or ‘OW’ in one case). One of these modified versions of REGE, REGGE-OW uses the same definition of regular equivalence as homogeneity (if the maximum is used as function \( f \)) and implicit blockmodeling, which was also called \( max \)-regular equivalence. The original and modified versions of REGE can be classified in four groups based on whether or not they treat the two possible ties between two units jointly or separately and based on whether they demand that all unique tie values
between evaluated units and a cluster or only the maximal ones are the same. The classification is presented in Table 4.1. A further distinction can be made among versions that require that all unique tie values between the evaluated units and a cluster must be equal based on whether they produce a measure of similarity (REGGE-OC, REGGE-OCOW) or a measure of dissimilarity (REGDI, REGDI-OW).

To more easily understand in what kind of networks they find partitions that perfectly correspond to the equivalences that these algorithms used they were applied to several artificial networks specially constructed for that purpose in Section 9.2.

Table 4.1: Classification of versions of REGE

<table>
<thead>
<tr>
<th>How are two possible ties between a pair of units treated:</th>
<th>Which tie values between the evaluated units and a cluster must be equal:</th>
</tr>
</thead>
<tbody>
<tr>
<td>all unique</td>
<td>maximal</td>
</tr>
<tr>
<td>jointly</td>
<td>REGDI, REGGE-OC</td>
</tr>
<tr>
<td>separately</td>
<td>REGDI-OW, REGGE-OCOW</td>
</tr>
<tr>
<td></td>
<td>REGGE</td>
</tr>
<tr>
<td></td>
<td>REGGE-OW</td>
</tr>
</tbody>
</table>

By incorporating the ideas of Nordlund (2007, 61) for determining regular blocks in valued networks (where tie values are influenced by the ‘size’ of the units) in these versions of REGE new versions of REGE were developed. These new versions are essentially the same as those on which they were developed with the only difference being that they analyze row and column normalized matrices instead of the original ones. They are marked by adding ‘NM’ or ‘-NM’ to the labels of the algorithms from which they were derived.

4.5 Structural similarity

In their 2004 paper Brandes and Lerner introduced the notion of structural similarity as a relaxation of equitable partitions. In Subsection 3.2.3 it was noted that for binary networks equitable partitions are actually the same as exact coloration (Everett and Borgatti, 1994: 43). They relax the equivalence with similarity and equivalence relation with a projection (Brandes and Lerner, 2004: 187). However, it should be noted that the computed similarities are normalized similarities. In the discrete case this would mean that a similarity between two equivalent units is 1 divided by the number of units in their equivalence class.

Structural similarity is similarity in terms of some graph characteristics. This characteristic can be centrality, a similar position in terms of core versus periphery, cohesive groups, opposing groups etc. (Brandes and Lerner, 2004: 193-194). How the characteristic that is incorporated in the similarity is controlled for is discussed later.

In their 2004 paper they developed this notation only for undirected graphs. They extended the notion to directed graphs in an unpublished paper (Brandes and Lerner, 2005) they presented at the Sunbelt XXV conference. However, this procedure is more complicated and remains untested. Therefore, the procedure for undirected graphs is presented here based on Brandes and Lerner (2004), with a few remarks on directed networks appearing at the end.
The procedure is based on an eigenvector decomposition of an adjacency matrix or a matrix representing an undirected graph (or a single-relation network). The k eigenvectors are chosen to represent a projection into k dimensional space, corresponding to k clusters in classical blockmodeling. The choice of appropriate eigenvalues and corresponding eigenvectors is actually the most problematic part of the procedure and is discussed later.

The similarity is then computed as $P^TP$, where $P$ is a matrix of selected eigenvectors with dimensions $k \times n$, $k$ is the selected number of eigenvectors and $n$ is as usual the number of units. As mentioned, the obtained similarities are normalized in the way that was explained above.

Structural similarities can be used as an input for some clustering procedure (as with all methods for indirect blockmodeling) and thus as a base for indirect blockmodeling. The problem that could arise here is that these similarities are normalized and can sometimes also have negative values. Another option for determining a partition would be to use raw selected eigenvectors as an input to an appropriate clustering method (such as k-means) or to compute a dissimilarity measure (such as Euclidian distance) on selected eigenvectors and use this as an input for an appropriate clustering method (such as some method of hierarchical clustering). The two methods (raw vectors and similarity) do not necessarily produce the same results. However, this contradicts the basic idea of Brandes and Lerner (2004: 185) who argue that ‘we need a notion of similarity of vertices (units), rather than equivalence’.

As already mentioned, the most difficult phase of this procedure is the selection of eigenvalues and corresponding eigenvectors. The selection of eigenvalues determines the structural characteristics of the graph that the similarity and eigenvectors represent. Brandes and Lerner (2004: 193-194) provide two rules for selecting eigenvalues. If searching for $k$-cohesive groups, select the $k$ largest eigenvalues. If searching for a core/periphery structure, select the largest and smallest eigenvalues. If searching for a core/periphery structure, select the largest and smallest eigenvalues.

Although at the beginning of the article they introduce structural similarity as a relaxation of equitable partitions, it should be emphasized that there is no (exact) procedure to obtain an equitable partition (or exact coloration) or a measure of (where an exact fit does not exist) equitable partition. If exact coloration exists, it can be obtained by selecting suitable eigenvalues and corresponding eigenvectors, yet there is no rule as to which eigenvalues to select.

For directed graphs, Brandes and Lerner (2005) propose a similar procedure. Again, the matrix is decomposed, however now a Schur decomposition is used. Appropriate Schur vectors are selected based on eigenvalues. Two new types of similarity are introduced: in-structural and out-structural similarity. Where a similarity is both in- and out-structural we can say it is structural. These similarities have not yet been tested and some guidelines

---

36 One should be positive and the other negative. The negative one should have an absolute value smaller than the positive one. If not, use the second smallest (still negative) one.
suggested for undirected graphs may not hold. In their 2005 paper Brandes and Lerner also study the stability and significance of structural similarities.

Structural similarities represent an interesting new family of similarities in terms of the structural characteristics of units in a network. However, there is a need for a clearer understanding of their meaning. This consideration is even more relevant in the version for directed networks. They could be used within the indirect blockmodeling approach, although this contradicts the basic idea of their authors (Brandes and Lerner, 2004: 185) who argue that similarities are more useful than equivalences.

4.6 Conclusion

This chapter mainly serves as a review of the indirect approaches. It first reviews CONCOR (Breiger et al., 1975), which is not a classical indirect approach. This is followed by a description of a procedure that characterizes the indirect approach and of the (dis)similarities most often used within this approach.

The section on regular equivalence (Section 4.4) goes beyond just a review of existing approaches and methods. Several modifications to versions of the REGE algorithm (algorithms for measuring (dis)similarities in terms of regular equivalence) are presented there. These modifications were performed for two purposes. The first is to make versions of REGE compatible (or at least more compatible) with definitions of regular equivalence (discussed in Section 3.3.2) that are used in generalized blockmodeling. The second is to adapt REGE to special kinds of networks where the tie values are heavily influenced by the ‘size’ (in some sense) of the units. Examples of such networks are trade networks among countries, networks of carbon flow among species, etc.

At the end, structural similarities (Brandes and Lerner, 2004) are presented as an interesting new family of similarities. However, there is a need for a clearer understanding of their meaning. This consideration is even more relevant in the version for directed networks.
5 DIRECT APPROACHES TO BLOCKMODELING

The definition of the direct approaches used here is taken from Batagelj et al. (1992b: 66) and was presented at the end of Chapter 3 (Introduction to blockmodeling). Direct approaches directly search for a partition of the network that best fits the selected equivalence. The fit of the partition to the selected equivalence is measured by a selected criterion function.

The selected criterion function should be computed using only network data (and the partition it evaluates) and should measure the extent to which a selected partition of units in a given network corresponds to the selected equivalence or other desired characteristics.

In the following section the history and basic characteristics of the direct approaches are described along with several examples from the related literature. Two approaches that fall into this class of blockmodeling approaches are reviewed in their own sections. The first is the approach introduced by Breiger and Mohr (2004). This is an approach to blockmodeling according to structural equivalence in contingency tables based on log-linear models. It is given special attention as a new approach to blockmodeling of valued networks.

The second approach that is reviewed in its own section is the generalized blockmodeling approach. While this approach is more thoroughly introduced in the following chapter, this section is used to introduce the generalized blockmodeling approach and position it within direct approaches to blockmodeling.

5.1 Introduction to direct approaches

The direct approach to blockmodeling was first introduced in three papers in the journal Social Networks 14 (Batagelj et al., 1992b; Batagelj et al., 1992a; Borgatti and Everett, 1992b). The first one (Batagelj et al., 1992b) introduced direct methods for structural equivalence and compared them with indirect methods, while the second one (Batagelj et al., 1992a) introduced methods for regular equivalence. The third paper (Borgatti and Everett, 1992b) did not focus on direct methods, yet it did introduce an alternative measure of fit for structural blockmodels that can also be applied to valued networks (Borgatti and Everett, 1992b: 101).

These early uses of indirect approaches were aimed at obtaining a better solution than could be obtained using indirect methods or at least getting some assurance that the obtained partition is at least locally optimal. Since direct methods incorporate a criterion function into the procedure searching for the optimal partition, we can be certain that if we check all possible partitions that the optimal partition/s\textsuperscript{37} will be found. No such guarantee is offered by

\textsuperscript{37} There may be more than one optimal partition.
indirect methods. However, due to the large\(^{38}\) number of possible partitions, a full or exhaustive search is usually impossible. Therefore, indirect methods usually either only optimize (try to improve) an existing partition (for example, one obtained using indirect methods) or use some search algorithm for searching through a space of possible partitions\(^{39}\). Such methods are:

- a local search with several (usually randomly selected) starting partitions (this is the procedure used in the methods proposed here, although any other could also be used);
- simulating annealing (Kirkpatrick et al., 1983);
- Tabu search (Glover, 1989); or
- genetic algorithm (Goldberg, 1989).

However, the incorporated measure of fit and possibility of finding better partitions are only some of the advantages of direct methods. In these early uses of direct approaches, a solution with the same properties (usually structural or regular equivalence) as those that could be found using indirect methods was sought. The second advantage lies in the flexibility of direct methods. Direct methods can be developed to find a partition with very specific properties. All that is needed is a criterion function that measures how similar (or how different) a certain partition is to an ideal solution or to the properties of the solution sought. Then some search procedure can be used to find a partition that hopefully\(^{40}\) maximizes (or minimizes) the value of the selected criterion function.

For example, Freeman (1993) tried to find clusters that correspond to the Homans (1950: 84 in Freeman, 1993: 227) definition of a group. He therefore constructed a criterion function (that he called ‘fitness’) that he believed best corresponds to that definition of a group (best measures the fitness of a partition) and then used a simple genetic algorithm to find a partition which maximizes that criterion function.

Similarly, Borgatti and Everett (1999) used another direct method to find core/periphery structures. In a way they produced the desired image matrix (or graph) and then searched for a partition which best matches that image. They used the correlation between the expected values based on the image matrix (and evaluated partition) and the observed values as a criterion function. One of the search algorithms described above can then be used to try to find the partition with the maximal value of the criterion function.

\(^{38}\) The number of partitions is Stirling number of the second kind (Ferligoj, 1989: 58) and rises very quickly with the number of units and the number of clusters. For example, there are 511 possible partitions of 10 units into 2 clusters, 9330 possible partitions of 10 units into 3 clusters and 16383 possible partitions of 15 units into 2 clusters. If we increase the number of units to 30 and the number of clusters to 4 (still a very small network and a relatively low number of clusters), the number of possible partitions increases to approximately \(4.8 \times 10^{16}\).

\(^{39}\) Which does not check all possible partitions, however there is a good (or at least significantly higher than random) chance that the optimal partition is checked.

\(^{40}\) Since an exhaustive search is usually infeasible, we can only hope that the obtained partition is optimal or at least close to the optimal partition.
5.2 Structural equivalence for contingency tables

Breiger and Mohr (2004) introduced methods based on log-linear models for evaluating the structural equivalence of counted data, which is for valued networks where the values of the ties represent counts. The base of the model is a log-linear model:

\[ \ln F_{ij} = u + u_i^R + u_j^C + u_{ii}^D + u_{ij}^{RC}, \]

where:
- \( F_{ij} \) are the estimated cell counts, with subscripts \( i \) and \( j \) indexing rows and columns,
- \( u \) is an intercept term or general effects,
- \( u_i^R \) and \( u_j^C \) denote row and column effects,
- \( u_{ii}^D \) diagonal interactions and \( u_{ij}^{RC} \) off-diagonal interactions.

If the \( u_{ii}^D \) and \( u_{ij}^{RC} \) terms are set to zero, then the model would simplify to the model of simple independence. This model becomes a model of structural equivalence for counted data when we impose the following restrictions (Bian et al., 2005):

\[ D_i = D_j \text{ for } i \in \text{cluster } C_k, \]
\[ R_{ij} = R_{lk} \text{ for } (i, j) \in \text{block } R(C_k, C_l). \]

Other restrictions are also possible for the evaluation of special cases of structural equivalence. For examples, see Breiger and Mohr (2004, 24-26). In their paper Breiger and Mohr (2004, 24) suggest likelihood-ratio \( \chi^2 \) statistics as a measure of fit. With this criterion function \( \chi^2 \) statistics, several partitions can be compared and evaluated in terms of structural equivalence of the counted data. To find an optimal partition, the same methods suggested earlier in the chapter on direct methods in general can be used.

However, Breiger and Mohr (2004, 30-35) also propose an agglomeration hierarchical method for clustering units in terms of structural equivalence. The algorithm described in the paper is developed for two-mode networks, however, the authors also developed a similar algorithm for one-mode networks. The algorithm in each stem joins the two units that produce a model that is the least significantly (has the largest p value) worse than the previous model.

5.3 Generalized blockmodeling

Batagelj, Doreian and Ferligoj took another approach. Instead of developing a specialized criterion function to solve specific problems, they developed a framework that can be used to solve a number of problems similar to those presented above. In Doreian et al. (1994) they introduced the concept of generalized equivalence and several block types. First briefly Batagelj (1997) and then more thoroughly in Batagelj et al. (1998) they introduced pre-specified blockmodeling which can be used to solve a wide range of problems. This

\footnote{A similar restriction can be found in Breiger and Mohr (2004). The only difference is that the diagonal elements are free in Breiger and Mohr (2004) and restricted to be equal within blocks.}
includes finding a core/periphery structure as in Borgatti and Everett (1999), cohesive subgroups (although with a slightly different definition than used in Freeman, 1993), hierarchical structures (for example, see Batagelj et al., 1998: 203-205), symmetric-acyclic decomposition (Doreian et al., 2000) and other problems. They named their approach Generalized Blockmodeling and recently published a book with the same title (Doreian et al., 2005) which gives an extensive overview. This approach provides the foundations for the main part of my thesis, new approaches to generalized blockmodeling of valued networks, and is therefore presented separately in the next chapter.

5.4 Conclusion

In this chapter, the history and basic characteristics of directed approaches were reviewed. Several approaches from the literature that fall into this category were also identified. Two of them were given special attention.

One of them is the approach presented by Breiger and Mohr (2004) for finding structural equivalence in a contingency table. It was specially considered here as it is a new approach that is suited for a special kind of data. I also believe that an idea used in this approach could be useful for developing other approaches to the blockmodeling of valued networks, also with some characteristics of generalized blockmodeling. Nevertheless, this option is not explored in this thesis.

The second direct approach that was introduced in this chapter is generalized blockmodeling. As this is the main focus of my thesis, this approach is presented in the following two chapters. In Chapter 6, generalized blockmodeling as developed by Doreian, Batagelj and Ferligoj is reviewed. My contributions to this approach in the form of its extensions to valued networks are presented in Chapter 7.
6 GENERALIZED BLOCKMODELING

Generalized blockmodeling was already introduced in a previous chapter in Section 5.3. This chapter intends to provide an overview of the existing state of generalized blockmodeling. In addition, some concepts that are needed to position the new approaches presented in the following chapter are introduced. The new approaches and other developments that result from my own studies in the field of generalized blockmodeling of valued networks are contained in Chapter 7.

6.1 Introduction to generalized blockmodeling

Doreian et al. (2005: 25-26) state that there are three main characteristics of generalized blockmodeling (in comparison to what they call conventional blockmodeling):

- the direct approach is an optimizational one (the algorithm works directly with network data and does not transform them into some other form);
- a much broader set of block types is used to define an equivalence instead of a few equivalence types; and
- the model can be pre-specified (not only the allowed block types, but also their positions).

Since in addition to their approach to generalized blockmodeling, several others are presented in this thesis there is a need to differentiate among them. Therefore, the notion of a type of (generalized) blockmodeling is introduced. Several types of generalized blockmodeling are distinguished: binary blockmodeling, valued blockmodeling, homogeneity blockmodeling and implicit blockmodeling. In the case of homogeneity blockmodeling, two subtypes are defined: sum of squares blockmodeling and absolute deviations blockmodeling. The aim of the dissertation is to discuss the blockmodeling of valued networks. Therefore, binary blockmodeling is considered only as a basis for developing appropriate approaches to the blockmodeling of valued networks. It is also used for a comparison with the proposed approaches.

The most important difference between the four main types of blockmodeling lies in an appropriate definition of the inconsistencies of the empirical blocks with the ideal ones. The way inconsistencies are computed have to consider the fact that the values on the ties in a

---

42 The term ‘generalized’ is usually omitted since all blockmodeling types referred to in the dissertation are generalized blockmodeling types.

43 The term binary blockmodeling type is used for the generalized blockmodeling of binary networks, which was thoroughly presented by Doreian et al. (2005).
valued network are measured on at least an interval scale. Therefore, ideal blocks must also be redefined. This is discussed in the following chapter.

Binary blockmodeling is presented in Section 6.3. It only analyzes binary networks and its criterion function measures block inconsistencies in principle by the number of errors. The other three main types analyze valued networks. They are mainly presented in the next chapter, although the ideas of Batagelj and Ferligoj (2000: 11-13) for implicit blockmodeling are already presented in Section 6.4 of this chapter. The criterion function of valued blockmodeling measures block inconsistencies as the deviation of appropriate values from either 0 or the value of the parameter $m$. The criterion function of implicit blockmodeling is similar, however, the value of parameter $m$ is set to the maximum of the block (or sometimes only of an analyzed row or column inside that block) analyzed. The criterion functions of both homogeneity blockmodeling types measure the block inconsistencies by the variability of the appropriate values. What these appropriate values are depends on the ideal block to which the inconsistencies are computed, however they are always computed based on the cell values of the block in question.

### 6.2 Criterion function

A common component of all types of blockmodeling is a basic criterion function. As mentioned in Section 3.5, the criterion function used in blockmodeling is a function that evaluates a partition based on network data and returns a value. In generalized blockmodeling the criterion function can only take on non-negative values. The partition with the lowest value of the criterion function is the best one while the value zero of the criterion function indicates that the partition perfectly matches the desired equivalence.

The only part of the criterion function that changes among different types of blockmodeling is that part where inconsistencies against ideal blocks are computed. In this section the part of the criterion function (only inconsistencies from ideal blocks are excluded) that is common to all types of blockmodeling is presented. It was presented in Batagelj et al. (1992b: 79) and Batagelj et al. (1992a: 124) and then extended in Doreian et al. (1994: 7-8). A full description can also be found in Doreian et al. (2005: 185-187, 223-226).

Let us first repeat some definitions and notations (for network, partition and block) and define the notation of an ideal block:

- The network $N = (U, R)$, where $U$ is a set of all units $U = (u_1, u_2, \ldots, u_n)$ and $R$ is the relation between these units $R \subseteq U \times U$

---

44 What the appropriate values are is determined by the ideal block from which inconsistencies for the selected empirical block are computed. These values are always based on the tie values in the selected empirical block.
In generalized blockmodeling, a relation \( R \) is usually represented by a matrix \( R \) with elements \( [r_{ij}] \), where \( r_{ij} \) indicates the value (or weight) of the arc from unit \( i \) to unit \( j \).

- \( C = \{ C_1, C_2, \ldots, C_n \} \) is a partition of the set \( U \). \( \Phi \) is a set of all feasible partitions. A partition \( C \) also partitions the relation \( R \) into (empirical) blocks \( R(C_i, C_j) = R \cap C_i \times C_j \). Each such block consists of units belonging to clusters \( C_i \) and \( C_j \) and all arcs leading from cluster \( C_i \) to cluster \( C_j \). If \( i = j \), the block \( R(C_i, C_i) \) is called a diagonal block.

- Let \( T(C_i, C_j) \) denote a set of all ideal blocks (of all block types allowed), corresponding to block \( R(C_i, C_j) \).

- \( w(T) \) is a weight given to block type \( T \). Usually, they are all set to 1, however, different weights can be given to different block types.

Block type inconsistency \( \delta (R(C_i, C_j), T) \) measures the inconsistency (deviation) of the empirical block \( R(C_i, C_j) \) with (from) an ideal block \( T \in T(C_i, C_j) \). This term can also be normalized to exclude the effect of block size by dividing it by the number of cells in the block.

Block inconsistency \( p(C_i, C_j) \) is defined as:

\[
p(C_i, C_j) = \min_{T \in T(C_i, C_j)} \left( w(T) \delta (R(C_i, C_j), T) \right).
\]

The total inconsistency \( P(C) \) of a partition \( C \) can be expressed as the sum of inconsistencies within each block (block inconsistencies) across all blocks:

\[
P(C) = \sum_{C_i, C_j \in C} p(C_i, C_j).
\]

The criterion function \( P(C) \) also possesses the following properties:

1. \( P(C) \geq 0 \)
2. \( P(C) = 0 \) if and only if \( C \) perfectly matches the selected equivalence defined through allowed block types and optionally their positions in the blockmodel.

The second property is called the sensitivity of a criterion function. If a criterion function is not sensitive, the 0 value of criterion function does not necessarily indicate that a network and a partition perfectly comply with the selected equivalence.

These definitions hold for all types of (generalized) blockmodeling. The difference between different blockmodeling types lies in the descriptions of ideal blocks of different types and in the definitions of block type inconsistencies for these block types. This is covered in Section 6.3 for binary blockmodeling and for the other types in Chapter 7.

To clarify the definition and usage of notions of block type, ideal block, empirical block, block type inconsistency and block inconsistency in this dissertation, the following explanation is given.

A block type is a type of connection between two clusters (or within a cluster). It is defined by a description of the pattern of ties and for valued networks also the tie values that are present in the ideal block of that type. Therefore, an ideal block of a given type is a block that perfectly matches that description of a block type. As the ideal block of a given type defines that block type, the terms are sometimes used interchangeably. An empirical block of a given
type is an empirical block that was classified as being that type when applying (generalized) blockmodeling.

Different block types are null, complete, regular etc. A block type with a given name is referred to as name block type, where name is substituted by the name of the block type in question. The word type is sometimes omitted so that name block type is only referred to as name block. However, name block is also used to refer to an empirical block classified as a block of the type name. The distinction should be clear from the context. Whenever this is not the case, the block type of a given name should be referred to as the name block type. An ideal block of a type of a given name is referred to as a name ideal block. To clarify the relationships between the notions of block type, ideal block and empirical block, a schema is presented below.

Table 6.1: Schema of ideal and empirical block types

<table>
<thead>
<tr>
<th>Block type</th>
<th>Ideal</th>
<th>Empirical</th>
</tr>
</thead>
<tbody>
<tr>
<td>null block (type)</td>
<td>null ideal block</td>
<td>(empirical) null block</td>
</tr>
<tr>
<td>complete block (type)</td>
<td>complete ideal block</td>
<td>(empirical) complete block</td>
</tr>
<tr>
<td>regular block (type)</td>
<td>regular ideal block</td>
<td>(empirical) regular block</td>
</tr>
<tr>
<td>name block (type)</td>
<td>name ideal block</td>
<td>(empirical) name block</td>
</tr>
</tbody>
</table>

The inconsistency of an empirical block $R(C_i, C_j)$ with a block type, or more precisely with an ideal block $T \in T(C_i, C_j)$, that is with an ideal block of that block type, is referred to as block type inconsistency. When referring to the inconsistency of an empirical block with an ideal block of a type name, the term name block inconsistency is used. For example, the inconsistency of an empirical block $R(C_i, C_j)$ with a null block type is termed null block inconsistency. The minimal block type inconsistency for a given empirical block is block inconsistency defined as

\[
p(C_i, C_j) = \min_{T \in T(C_i, C_j)} \delta(R(C_i, C_j), T).
\]

6.3 Binary blockmodeling

Generalized blockmodeling was first suggested and applied to binary networks (Batagelj et al. 1992a; Batagelj et al., 1992b; Doreian et al., 1994). Also, most of the book by Doreian et al. (2005) on generalized blockmodeling focuses on binary networks. Therefore, this area is the most developed one.

As mentioned, a criterion function described in the previous section is used in all types of generalized blockmodeling, including the binary blockmodeling type. However, only that part of the criterion function common to all types of blockmodeling was presented in the previous

\[\text{name can be replaced by the name of any block type.}\]
section. In order to complete the criterion function for binary blockmodeling, ideal blocks and corresponding block type inconsistencies must be specified.

Ideal blocks for binary blockmodeling are described in Table 6.2.

*Table 6.2: Description of ideal blocks for block types (Doreian et al., 2005: 223)*

<table>
<thead>
<tr>
<th>Ideal block</th>
<th>With ‘label’</th>
<th>Descriptions of ideal blocks (of a given type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>‘null’</td>
<td>All 0 *</td>
</tr>
<tr>
<td>Complete</td>
<td>‘com’</td>
<td>All 1 *</td>
</tr>
<tr>
<td>Row-dominant</td>
<td>‘rdo’</td>
<td>$\exists$ all 1 row *</td>
</tr>
<tr>
<td>Col-dominant</td>
<td>‘cdo’</td>
<td>$\exists$ all 1 column *</td>
</tr>
<tr>
<td>Row-regular</td>
<td>‘rre’</td>
<td>1-covered rows</td>
</tr>
<tr>
<td>Column-regular</td>
<td>‘cre’</td>
<td>1-covered columns</td>
</tr>
<tr>
<td>Regular</td>
<td>‘reg’</td>
<td>1-covered rows and 1-covered columns</td>
</tr>
<tr>
<td>Row-functional</td>
<td>‘rfn’</td>
<td>$\exists!$ one 1 in each row</td>
</tr>
<tr>
<td>Column-functional</td>
<td>‘cfn’</td>
<td>$\exists!$ one 1 in each column</td>
</tr>
<tr>
<td>Density</td>
<td>‘den’</td>
<td>Number of 1s $\geq \gamma \cdot$ number of cells in the block</td>
</tr>
</tbody>
</table>

Legend:

* - an exception may be cells on the diagonal where either all cells should have a value of 0 or all a value of 1

$\gamma$ – a pre-specified parameter for minimum density in density blocks.

In Table 6.3, block inconsistencies for binary blockmodeling are defined. The formulas in that table are for computing the criterion function based on a binary network. If the network is valued it has to be binarized beforehand. This is done by recoding all values below some pre-specified threshold ($t$) into 0s and all others into 1s.
Table 6.3: Computation of block type inconsistencies for binary blockmodeling (slightly adapted from Doreian et al., 2005: 224)

<table>
<thead>
<tr>
<th>Block type</th>
<th>block type inconsistencies - ( \delta(R(C_i, C_j), T) )</th>
<th>Position of the block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>( s_t )</td>
<td>off-diagonal</td>
</tr>
<tr>
<td></td>
<td>( s_t + \min(0, n_t - 2s_d) )</td>
<td>diagonal</td>
</tr>
<tr>
<td>Complete</td>
<td>( n_t n_c - s_t )</td>
<td>off-diagonal</td>
</tr>
<tr>
<td></td>
<td>( n_t n_c - s_t + \min(- n_t + 2s_d, 0) )</td>
<td>diagonal</td>
</tr>
<tr>
<td>Row-dominant</td>
<td>( (n_c - m_r)n_r )</td>
<td>off-diagonal</td>
</tr>
<tr>
<td></td>
<td>( [n_c - m_r - \text{usepos}(1 - s_d)]n_r )</td>
<td>diagonal</td>
</tr>
<tr>
<td>Column-dominant</td>
<td>( (n_r - m_c)n_c )</td>
<td>off-diagonal</td>
</tr>
<tr>
<td></td>
<td>( [n_r - m_c - \text{usepos}(1 - s_d)]n_c )</td>
<td>diagonal</td>
</tr>
<tr>
<td>Row-regular</td>
<td>( (n_r - p_r)n_c )</td>
<td></td>
</tr>
<tr>
<td>Column-regular</td>
<td>( (n_c - p_c)n_r )</td>
<td></td>
</tr>
<tr>
<td>Regular</td>
<td>( (n_c - p_c)n_r + (n_r - p_r)p_c )</td>
<td></td>
</tr>
<tr>
<td>Row-functional</td>
<td>( s_t - p_r + (n_r - p_r)n_c )</td>
<td></td>
</tr>
<tr>
<td>Column-functional</td>
<td>( s_t - p_c + (n_c - p_c)n_r )</td>
<td></td>
</tr>
<tr>
<td>Density</td>
<td>( \max(0, \gamma n_t n_c - s_t) )</td>
<td></td>
</tr>
</tbody>
</table>

Legend:
\( s_t \) total block sum = number of 1s in a block
\( s_d \) diagonal block sum = number of 1s on a diagonal
\( n_t \) number of rows in a block = card \( C_i \)
\( n_c \) number of columns in a block = card \( C_j \)
\( n_r \) number of non-null rows in a block
\( p_c \) number of non-null columns in a block
\( m_r \) maximal row-sum
\( m_c \) maximal column-sum
\( \gamma \) a pre-specified parameter for minimum density in density blocks.

6.4 Existing ideas for valued networks

Batagelj and Ferligoj (2000: 12-13) presented a direct approach for clustering relational data that they called an *implicit approach*. They discussed this approach in the context of generalized blockmodeling and clustering with relational constraints, which they see as special cases of the clustering relational data problem. They keep the original criterion function as described in Section 6.2 (Criterion function). Their approach can be seen as a
generalized blockmodeling approach for valued matrices (with only non-negative values). The term network is deliberately not used here since the procedure is designed for matrices that are the result of relational and attribute data. However, valued networks can be represented by valued matrices. Similarly to other types of generalized blockmodeling, the only difference between this type and other types lies in the definitions and descriptions of the ideal blocks of selected block types and, consequently, in the way block type inconsistencies of empirical blocks with these ideal blocks are computed. In their paper, they do not provide any descriptions or definitions of ideal blocks, just three examples of formulas for computing inconsistencies.

Table 6.4: The rewritten original examples of Batagelj and Ferligoj (2000: 13)

<table>
<thead>
<tr>
<th>Block types</th>
<th>block type inconsistencies - ( \delta(R(C_i, C_j), T) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>( \frac{\sum B}{n_c \cdot n_c \cdot \max{B \neq 0}} )</td>
</tr>
<tr>
<td>Row-dominant</td>
<td>( 1 - \max \frac{\sum B_{[i,j]}}{n_c \cdot \max{B_{[i,j]} \neq 0}} )</td>
</tr>
<tr>
<td>Column-(max-(46))regular</td>
<td>( 1 - \frac{\sum \max B_{[i,j]}}{n_c \cdot \max{B \neq 0}} )</td>
</tr>
</tbody>
</table>

If the max in the denominator equals 0 the fraction has a value of 0.

The three original formulas are written in Table 6.4 with a different notation. Obviously, these inconsistencies are already normalized to take a value between 0 and 1. Here Batagelj and Ferligoj stopped in the presentation of their ideas. I develop this approach further in Section 7.3.

6.5 Conclusion

In this chapter, the fundamental characteristics of generalized blockmodeling were presented. The type of criterion function described in Section 6.2 is used in all generalized blockmodeling approaches. The main idea of this type of criterion function is that inconsistencies of a selected partition (based on a specific network) with an equivalence are measured as the sum of inconsistencies of individual empirical blocks with their ideal blocks. What kinds of ideal blocks can be used is specified by the equivalence, which can also be generalized equivalence.

\[46\] In the article by Batagelj and Ferligoj (2000: 13), the block is only called column-regular. However, for clarity reasons (see Subsection 3.3.3), the ‘max-’ is added here.
The ideal blocks for binary blockmodeling presented in Section 6.3 are, together with the previously mentioned criterion function (presented in Section 6.2), the foundations on which the approaches presented in Chapter 7 were built. Although mainly used for further development of the implicit approach (in Subsection 7.3), the ideas for the blockmodeling of valued networks presented by Batagelj and Ferligoj (2000: 11-13) are also closely related to valued blockmodeling developed in Section 7.1.
7 NEW APPROACHES TO THE GENERALIZED BLOCKMODELING OF VALUED NETWORKS

This chapter represents the core of the dissertation. Here, the new approaches or blockmodeling types for the generalized blockmodeling of valued networks announced in the previous chapter are developed. These are valued, homogeneity and implicit blockmodeling.

Valued blockmodeling is developed in Section 7.1 based on binary blockmodeling. Homogeneity blockmodeling is then developed in Section 7.2 based on the idea that block inconsistency in a valued network should be measured by some measure of variability. Similar ideas for structural equivalence were already presented by Borgatti and Everett (1992b: 101), where they stated that the average variance within matrix blocks is a natural measure of fit. The bases for implicit blockmodeling were the ideas of Batagelj and Ferligoj (2000: 11-13), which they called an implicit approach to clustering relational data. These ideas were already presented in Section 6.4. In Section 7.3, these ideas are further developed into a full blockmodeling type with a wide set of possible block types. In addition, some modifications are suggested that seem to improve its performance on empirical networks and address some theoretical issues.

This is followed by a presentation of the rules for determining the values of connections in reduced graphs and some suggestions for possible improvements or just alternative versions of the suggested approaches. At the end of the chapter the proposed approaches are compared from the theoretical prospective with indirect approaches and among themselves.

7.1 Generalization of binary blockmodeling to valued blockmodeling

The approach presented in this section was inspired by the fact that in the past, when valued networks were analyzed using generalized blockmodeling, they were first converted into binary networks in such a way that values greater than (or equal to) a certain threshold were recoded into 1s and the other into 0s (examples can be found in Doreian et al., 2005: 35-37, 44-47, 54-60, 265-269). Analyzing a network in such a way causes the loss of a considerable amount of information. The valued blockmodeling reduces the amount of information lost, although a loss usually still occurs. Information about the values of ties (or sometimes the values of function $f$ on certain values) above parameter $m$ (a parameter that is defined later) is lost. If $m$ is set to a sufficiently high value, no information is lost. However, this might not...
be appropriate for many networks since it might cause almost all or even all empirical blocks to be classified as null.

Due to the loss of information, both binary and valued blockmodeling\(^\text{48}\) are not sensitive\(^\text{49}\) to the departures from the definitions presented in Section 3.3 (Equivalences for valued networks). The exception is \(f\)-regular equivalence at level \(m\), which was defined with valued blockmodeling in mind. Therefore, the valued blockmodeling is sensitive to departures from this definition. Of course, other definitions can be modified in a similar way so that the valued blockmodeling would also be sensitive to departures from them. Similarly, definitions can be modified so that the binary blockmodeling would be sensitive to them.

However, this is unnecessary. Dorian, Batagelj, and Ferligoj (1994) introduced generalized equivalence whereby equivalence is defined by set of allowed block types and optionally their positions in the blockmodel. Block types are defined separately for each blockmodeling type by the descriptions of corresponding ideal blocks. Both binary and valued blockmodeling are sensitive to departures from generalized equivalences defined by a set of allowed block types of the corresponding blockmodeling type and optionally their positions in the blockmodel.

The descriptions of ideal blocks for binary blockmodeling were already presented in Table 6.2. For valued blockmodeling, they are presented in this section. However, let us first look again at the descriptions of ideal blocks for binary networks. They are again presented in Table 7.1 (for now, look only at the first two columns) as they also represent the basis upon which the descriptions of ideal blocks for valued blockmodeling were developed.

To better understand the generalizations it is useful to divide the ideal block into two groups:

- **Group 1**: Those ideal blocks which demand that certain cells have a tie (1) and/or certain cells do not have a tie (0). This group includes blocks such as:
  - Complete: all cells must be 1
  - Row-dominant: all cells in at least one row must be 1
  - Column-dominant: all cells in at least one column must be 1
  - Row-functional: exactly one cell in each row must be 1, all others 0
  - Column-functional: exactly one cell in each column must be 1, all others 0
  - Null: all cells must be 0

- **Group 2**: The ideal blocks which demand that each row or column (or both) have at least one tie (1). This group includes all regular-like ideal blocks:
  - Row-regular: there must be at least one tie (1) in each row

---

\(^48\) Recoding a valued network into a binary one clearly causes a loss of information. The valued information is reduced to a binary one. Valued blockmodeling also suffers from a loss of information, although to a much smaller extent, unless the condition from the previous condition is satisfied.

\(^49\) Sensitivity as a property of the criterion function is described in Section 6.2.
Therefore, we can see that there are three types of conditions:
1. a certain cell must be (at least) 1;
2. a certain cell must be 0; and
3. at least 1 cell in each row (or column) must be 1 or, to put it differently, the \( f \) of all cells in each row (or column) must be at least 1, where \( f \) is some function over the values that has the property \( f(a) \geq \max(a) \), and \( a \) is a valued vector.

Table 7.1: Description of ideal blocks

<table>
<thead>
<tr>
<th>Ideal block with ‘label’</th>
<th>Description for binary blockmodeling</th>
<th>Description for valued blockmodeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null ‘null’</td>
<td>all 0 *</td>
<td>all 0 *</td>
</tr>
<tr>
<td>Complete ‘com’</td>
<td>all 1 *</td>
<td>all values at least ( m ) *</td>
</tr>
<tr>
<td>Row-dominant ‘rdo’</td>
<td>there exists an all 1 row *</td>
<td>there exists a row where all values are at least ( m ) *</td>
</tr>
<tr>
<td>Col-dominant ‘cdo’</td>
<td>there exists an all 1 column *</td>
<td>there exists a column where all values are at least ( m ) *</td>
</tr>
<tr>
<td>Row-(f-)regular ‘rre’</td>
<td>there exists at least one 1 in each row</td>
<td>the ( f ) over each row is at least ( m )</td>
</tr>
<tr>
<td>Column-(f-)regular ‘cre’</td>
<td>there exists at least one 1 in each column</td>
<td>the ( f ) over each column is at least ( m )</td>
</tr>
<tr>
<td>(f-)regular ‘reg’</td>
<td>there exists at least one 1 in each row and each column</td>
<td>the ( f ) over each row and each column is at least ( m )</td>
</tr>
<tr>
<td>Row-functional ‘rfn’</td>
<td>there exists exactly one 1 in each row</td>
<td>there exists exactly one tie with a value of at least ( m ) in each row, all others are 0</td>
</tr>
<tr>
<td>Column-functional ‘cfn’</td>
<td>there exists exactly one 1 in each column</td>
<td>there exists exactly one tie with a value of at least ( m ) in each column, all others are 0</td>
</tr>
<tr>
<td>Density ( \gamma ) ‘dns’</td>
<td>the number of 1s ( \geq \gamma \cdot \text{(number of cells)} )</td>
<td>the sum of all values ( \geq \mu \cdot \text{(number of cells)} )</td>
</tr>
<tr>
<td>Average ( \mu ) ‘avg’</td>
<td>( \gamma \cdot \text{(number of cells)} )</td>
<td>( \mu \cdot \text{(number of cells)} )</td>
</tr>
</tbody>
</table>

Legend:
* - an exception may be cells on the diagonal where either all cells should have a value of 0 or all a value of 1
\( \diamond \) - an exception may be cells on the diagonal where either all cells should have a value of 0 or all a value of at least \( m \)

\( 50 \) Doreian, Batagelj and Ferligoj, 2005: 223
At least one of these three conditions can be found in each (previously mentioned) ideal block and the three of them together are enough to define (with the correct specification of certain cells) any of those ideal blocks (and block types) and compute their block type inconsistencies. The inconsistency of an empirical block with a given ideal block is usually the weighted sum of errors for these conditions or, more precisely, the weighted sum of the number of times each condition is broken (which applies to a certain block). The term ‘weighted sum’ is used since sometimes the block inconsistency or part of it is multiplied by the number of cells in a given row or column.

Therefore, if we can generalize these three conditions to valued networks we have in fact generalized the criterion function and blockmodeling to valued networks. Only a slight modification is needed to generalize these three conditions to valued networks and that is to replace the ones (1s) in the conditions with an appropriate parameter, lets call it $m$. The new conditions can then be written as:

1. A certain cell must be (at least) $m$;
2. A certain cell must be 0; and
3. the value of function $f$ over all cells in each row (or column) must be at least $m$.

Based on the generalization of these conditions and descriptions of ideal blocks for binary networks (presented in the second column of Table 7.1), the description of ideal blocks for valued networks presented in Table 7.1 (the third column) can be derived. The only exception is the description of the density ideal block. Its description is not derived from the generalization of those conditions. However, the density ideal block from binary blockmodeling also has its counterpart in valued blockmodeling. This is the average ideal block.

Mainly on the basis of these descriptions we can now construct formulas for computing deviations (block type inconsistencies) from these ideal blocks. In Table 7.2 the computations of block type inconsistencies for valued blockmodeling (and for comparison for binary blockmodeling) for all block types are presented. Another nice property of this generalization is that if we have a binary network and set $m$ to 1 and function $f$ to the maximum, we get the inconsistencies for binary networks. Therefore, binary blockmodeling can be seen as a special case of valued blockmodeling. Also, the null block inconsistency is the same as that part of the criterion function proposed by Doreian and Mrvar (1996: 161) corresponding to positive errors and confined to only one block.

If the $f$ is the maximum, then if in a certain situation (network, blockmodel or allowed block types, partition, $m$) the value of the criterion function is 0 using valued blockmodeling, the value of the criterion function would also be 0 if the network were converted into a binary

---

51 There is one exception, the density ideal block, which is not encompassed by these rules. The corresponding ideal block in valued blockmodeling is the average ideal block.

52 The only exception is the inconsistency for a regular block type.
network using threshold $t$ equal to $m$ and then analyzed using binary blockmodeling. The opposite is usually not true.

Table 7.2: Computation of block type inconsistencies for binary and valued (generalized from binary) blockmodeling

<table>
<thead>
<tr>
<th>Block type</th>
<th>Binary blockmodeling $\delta(R(C_a, C_b), T)$</th>
<th>Valued blockmodeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>$s_t$</td>
<td>$\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} b_{ij}$</td>
</tr>
<tr>
<td></td>
<td>$s_t + \min(0, n_r - 2s_d)$</td>
<td>$\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} b_{ij} + \min(0, \sum (m - \text{diag}(B))^+ - \sum \text{diag}(B))$</td>
</tr>
<tr>
<td>complete</td>
<td>$n_r n_c - s_t$</td>
<td>$\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} (m - b_{ij})^+$</td>
</tr>
<tr>
<td></td>
<td>$n_r n_c - s_t + \min(- n_r + 2s_d, 0)$</td>
<td>$\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} (m - b_{ij})^+ + \min(- \sum (m - \text{diag}(B))^+ + \sum \text{diag}(B), 0)$</td>
</tr>
<tr>
<td>row-dominant</td>
<td>$(n_c - m_r) n_r$</td>
<td>$\min((m - B)^+ \cdot 1) n_r$</td>
</tr>
<tr>
<td></td>
<td>$[n_r - m_r - (1 - s_d)^+] n_r$</td>
<td>$\min((m - B)^+ \cdot 1 + \sum \text{diag}(B) - \sum \text{diag}(m - B)^+ + \sum \text{diag}(B), 0) n_r$</td>
</tr>
<tr>
<td>column-dominant</td>
<td>$(n_r - m_c) n_c$</td>
<td>$\min((-1)(m - B)^+ \cdot 1) n_c$</td>
</tr>
<tr>
<td></td>
<td>$[n_r - m_c - (1 - s_d)^+] n_c$</td>
<td>$\min((-1)(m - B)^+ \cdot 1 + \sum \text{diag}(B) - \sum \text{diag}(m - B)^+ + \sum \text{diag}(B), 0) n_c$</td>
</tr>
<tr>
<td>row-regular</td>
<td>$(n_r - p_r) n_c$</td>
<td>$\sum_{i=1}^{n_r} (m - f(B_{i1}))^+ n_c$</td>
</tr>
<tr>
<td>column-regular</td>
<td>$(n_c - p_r) n_r$</td>
<td>$\sum_{j=1}^{n_c} (m - f(B_{1j}))^+ n_r$</td>
</tr>
<tr>
<td>$f$-regular</td>
<td>$(n_c - p_r) n_r + (n_r - p_r) p_c$</td>
<td>$\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} \max((m - f(B_{i1}))^+, (m - f(B_{1j}))^+)$</td>
</tr>
</tbody>
</table>

$^{53}$ Slightly adapted from Doreian et al. (2005: 224).
Block type inconsistencies - $\delta(R(C_a, C_b), T)$

<table>
<thead>
<tr>
<th>Block type</th>
<th>Binary blockmodeling</th>
<th>Valued blockmodeling</th>
<th>Position of the block</th>
</tr>
</thead>
<tbody>
<tr>
<td>row-functional $s_i - p_r +$</td>
<td>$(n_r - p_r)n_c$</td>
<td>$\sum_{i=1}^{n_r} \left( m - \max(B_{[i,j]}) \right) n_c + \sum_{j=1, j \neq \arg \max(B_{[i,j]})}^{n_c} B_{[i,j]}$</td>
<td></td>
</tr>
<tr>
<td>column-functional $s_i - p_c +$</td>
<td>$(n_c - p_c)n_r$</td>
<td>$\sum_{j=1}^{n_c} \left( m - \max(B_{[i,j]}) \right) n_r + \sum_{i=1, i \neq \arg \max(B_{[i,j]})}^{n_r} B_{[i,j]}$</td>
<td></td>
</tr>
</tbody>
</table>

Density $\gamma$ max$(0, \frac{1}{m} \sum_{i=1}^{n_r} \sum_{j=1}^{n_c} b_{ij})$

Average $\mu$ $\frac{\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} b_{ij}}{mn_r n_c}$

Legend:
- $s_i$ total block sum = number of 1s in a block
- $s_d$ diagonal block sum = number of 1s on a diagonal
- $n_r$ number of rows in a block = $\text{card } C_i$
- $n_c$ number of columns in a block = $\text{card } C_j$
- $p_r$ number of non-null rows in a block
- $p_c$ number of non-null columns in a block
- $m_r$ maximal row-sum
- $m_c$ maximal column-sum
- $B$ the matrix of the block $R(C_a, C_b)$
- $B_{[i,j]}$ the $i$-th row of matrix $B$
- $B_{[j,i]}$ the $j$-th column of matrix $B$
- $b_{ij}$ an element of matrix $B$ defined by the $i$-th row and $j$-th column
- $\text{diag}$ extract the diagonal elements of the matrix

### 7.1.1 Parameter $m$

The main problem is to determine the right value of parameter $m$, which presents the minimal value that must characterize the tie between a unit and either a cluster (for block types $f$-regular, row-$f$-regular and column-$f$-regular) or another unit (for block types complete, row-dominant, column-dominant, row-functional and column-functional) so that this tie is considered strong enough to fully comply with the description of a selected ideal block.

For example, let us consider a citation network and $m$ set to 5. A block $R(C_a, C_b)$ in such a network is an ideal complete block if every author from cluster $C_i$ cites at least 5 papers from every author from cluster $C_j$. Here, each author of cluster $C_i$ must be (strongly) connected to each author from cluster $C_j$. A block $R(C_a, C_b)$ in such a network is an ideal sum-regular block, if every author from cluster $C_i$ cites authors from cluster $C_j$ at least 5 times and every author from cluster $C_j$ is cited at least 5 times by the authors of cluster $C_i$.

It would be best if we could set $m$ according to some prior knowledge about the problem or at least from the network characteristics. A previous solution can also be useful for determining the correct $m$. Also, as we know the appropriate value or threshold $t$ for binarizing the
network (for use with binary blockmodeling), the value of $m$ should be approximately $m = 2t$
or lower.

If prior knowledge is unavailable, the empirical experience suggests that $m$ could be
determined using the distribution of cell values for models without $f$-regular, row- and
column-$f$-regular block types (e.g. models with complete, dominant and null block types) and
the distribution of row or column $f$-regular statistics for those with $f$-regular, row- or column-$f$-
regular block types. The exception is when function $f$ is maximum. In such cases, the
distribution or cell values can also be used for models with $f$-regular, row- and column-$f$-
regular block types.

In such cases, the suitable value of $m$ is usually about twice the value where the density of this
distribution is minimal. The value where the density of the distribution is minimal is also
usually close to the appropriate value of threshold $t$ for binarizing the network for use in
binary blockmodeling.

If the distribution of row or column statistics is examined, and the statistics used is influenced
by the number of units over which it is computed (such as the sum), the number of expected
empirical $f$-regular or row- or column-$f$-regular blocks we expect for at least some clusters
should be taken into account.

However, much better estimates (than those usually obtained following the suggestions in the
two previous paragraphs) of the appropriate value of $m$ can be obtained by examining some
prior partition that is close to optimal. These partitions could be obtained with the indirect
methods described earlier or with the homogeneity or implicit blockmodeling described in the
next two sections. There are two problems with this way of estimating the value of parameter
$m$. Firstly, the partition used in the estimation of $m$ can heavily influence the obtained
solution. This is not so disturbing since $m$ always (and therefore also the method for
estimating) heavily influences the solution. The second problem is that other methods
sometimes use different definitions of certain block types and some (indirect methods and
homogeneity blockmodeling) do not allow as rich blockmodels as valued blockmodeling,
which means that sometimes ‘prior partition close to optimal’ cannot be obtained.

---

54 If row-$f$-regular blocks are of the main interest then row statistics are relevant, while if column-$f$-regular
blocks are of main interest then column statistics are relevant. If $f$-regular block type or both (row- and column-$f$-
regular) are of interest then both row and column statistics are relevant. They should be examined either as
separate distributions, as one distribution with elements from both distributions or as a distribution of their
means.

55 Appropriate statistics should match the $f$ used in the definition of $f$-regular, row- and column-$f$-regular blocks.

56 Homogeneity blockmodeling does allow just as rich blockmodels as valued blockmodeling if block types that
are still experimental and those that have compatibility problems are also used.
7.2 Homogeneity blockmodeling

Another approach to blockmodeling that takes the tie values into account is to search for homogeneity within blocks. The homogeneity blockmodeling searches for the partition where the sum of some measure of variability within blocks over all blocks would be the lowest. This measure of variability within a block presents a block type inconsistency. This is basically enough to incorporate these new block type inconsistencies into the framework presented previously, that is in the criterion function presented in Section 6.2. Some measure of variability within blocks is a very broad term and has to be defined. So far, two measures of variability have been considered, the sum of square deviations from the mean (Sum of Squares) and the sum of absolute deviations from the median (Absolute Deviations). What remains to be established here is over which values these measures have to be computed. This depends on the block type.

This measure can either be computed over all cells in a block or over selections of cells that amount to all cells in the block (e.g., for complete and row- and column-functional block types), over only some selected cells in a block (e.g., over cells of only one row or column like in row- of column-dominant block types), or over function on rows or columns (e.g., for row-\(f\)-regular, column-\(f\)-regular and \(f\)-regular block types). If the measure is computed over the values of the function \(f\) of rows or columns, the result should then be multiplied by the number of elements in each row or column respectively. Also, if the measure is computed over only a selection of cell values the result should be divided by that proportion that the selected cell values represent in the whole block. In this way the block type inconsistencies for sum of squares complete, sum of squares row mean-regular, and sum of squares column-dominant block type match if the rows are homogeneous, that is if each row has zero variance.

The description of ideal blocks for homogeneity blockmodeling is provided in Table 7.3. Here it can be seen that (if the conditions in brackets are omitted) the null block type (in the homogeneity blockmodeling approach) is only a special case of the complete block type. As always with generalized blockmodeling, the complete block type is a special case of row-dominant, column-dominant, row-functional, column-functional, row-\((f)\)-regular, column-\((f)\)-regular and \((f)\)-regular block types. With homogeneity blockmodeling, the null block is therefore also a special case of these blocks. This can often be problematic as the distinction between the null block type and other block types is usually of great importance. The problem is addressed in more detail in Section 7.4.

Block type inconsistencies can be defined based on these descriptions and a selected measure of variability. The block type inconsistencies for both measures of variability (sum of square deviations from the mean and sum of absolute deviations from the median) are presented in Table 7.4. In Table 7.3, several blocks had the condition that the values that should be equal should also be different from zero. However, this condition is not reflected in the formulas for their block type inconsistencies. These inconsistencies are also 0 if all cells in a block have a value of 0. This problem is addressed in Section 7.4.
<table>
<thead>
<tr>
<th>Ideal block</th>
<th>Description for both homogeneity blockmodeling types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null ‘null’</td>
<td>All 0 (except may be diagonal, where all are then equal)</td>
</tr>
<tr>
<td>Complete ‘com’</td>
<td>All equal (and non-zero)* (diagonal may be treated separately)</td>
</tr>
<tr>
<td>Row-dominant version 1 ‘rdo1’</td>
<td>the row that is most different from the others has all values that are equal and (all values in a block are not zero)* (except may be diagonal, where all are then equal)</td>
</tr>
<tr>
<td>Col-dominant version 1 ‘cdo1’</td>
<td>the column that is most different from the others has all values that are equal (and all values in a block are not zero)* (except may be diagonal, where all are then equal)</td>
</tr>
<tr>
<td>Row-dominant version 2 ‘rdo2’</td>
<td>there exists a row where all values are equal (and all values in a block are not zero)* (except may be diagonal, where all are then equal)</td>
</tr>
<tr>
<td>Col-dominant version 2 ‘cdo2’</td>
<td>there exists a column where all values are equal (and all values in a block are not zero)* (except may be diagonal, where all are then equal)</td>
</tr>
<tr>
<td>Row-dominant version 3 ‘rdo 3’</td>
<td>the row with the highest central value has all values that are equal (and non-zero)* (except may be diagonal, where all are then equal)</td>
</tr>
<tr>
<td>Col-dominant version 3 ‘cdo 3’</td>
<td>the column with the highest central value has all values that are equal (and non-zero)* (except may be diagonal, where all are then equal)</td>
</tr>
<tr>
<td>Row-functional ‘rfn’</td>
<td>maximums (must be unique) of all rows are equal (and non-zero)*, all other values 0</td>
</tr>
<tr>
<td>Column-functional ‘cfn’</td>
<td>maximums (must be unique) of all columns are equal (and non-zero)*, all other values are 0</td>
</tr>
<tr>
<td>Row(-f)-regular ‘rre’</td>
<td>function $f$ on each row is equal (and non-zero)* for all rows</td>
</tr>
<tr>
<td>Column(-f)-regular ‘cre’</td>
<td>function $f$ on each column is equal (and non-zero)* for all columns</td>
</tr>
<tr>
<td>(f)-regular ‘reg’</td>
<td>function $f$ on each row and each column is equal (and non-zero)* for all rows and all columns separately</td>
</tr>
</tbody>
</table>

Legend:
* - the conditions in brackets are added to distinguish these blocks from null blocks. Although usually theoretically desired, the conditions are optional. The condition is not regarded unless explicitly stated and is not reflected in the way inconsistencies are computed in Table 7.4. The discussion of whether or not the condition should be included can be found in Section 7.4.
+ - The null block type is included in this list mostly for continuity reasons. Unless the conditions in brackets marked with * are used, it is not necessary as it is generally only a special case of all other block types. If the conditions marked with * are used, the inclusion of null block type is necessary.
Table 7.4: Block inconsistencies for homogeneity blockmodeling

<table>
<thead>
<tr>
<th>Ideal block</th>
<th>Block inconsistencies - $\delta(R(C_a, C_b), T)$</th>
<th>Position of the block</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>$\sum_{i,j} b_{ij}^2$ $</td>
<td>\sum_{i,j} b_{ij}</td>
</tr>
<tr>
<td>complete</td>
<td>$ss(b_{ij})$ $ad(b_{ij})$</td>
<td>diagonal</td>
</tr>
<tr>
<td>row-dominant version 1</td>
<td>$\min_{i=\text{argmax}} \left( \min_{j} \left( ss(b_{ij}) \right) \right) n_r$</td>
<td>diagonal</td>
</tr>
</tbody>
</table>
# Ideal block

## Block inconsistencies - $\delta(R(C, T))$

<table>
<thead>
<tr>
<th>Position of the block</th>
<th>Ideal block</th>
<th>Sum of squares</th>
<th>Absolute deviations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off-diagonal</td>
<td>$\min_{j=\text{argmax}} \left( \frac{ss(b_{ij})}{i} \right)<em>{j} h</em>{c}$</td>
<td>$\min_{j=\text{argmax}} \left( \text{Me}(b_{ij}) - \text{Me}(b_{jk}) \right)<em>{j} a</em>{d}(b_{ij}) h_{c}$</td>
<td></td>
</tr>
<tr>
<td><strong>Column-dominant</strong></td>
<td>$\min_{j=\text{argmax}} \left( \frac{ss(b_{ij})}{i} + (b_{ij} - \text{diag}(B))^{2} \right)<em>{j} n</em>{c}$</td>
<td>$\min_{j=\text{argmax}} \left( \text{Me}(b_{ij}) - \text{Me}(b_{jk}) \right)<em>{j} a</em>{d}(b_{ij}) h_{c}$</td>
<td></td>
</tr>
<tr>
<td><strong>Row-dominant</strong></td>
<td>$\min_{i=1} \left( ss(b_{ij}) \right)<em>{i} n</em>{r}$</td>
<td>$\min_{i=1} \left( ad(b_{ij}) \right)<em>{i} n</em>{r}$</td>
<td></td>
</tr>
<tr>
<td><strong>(Simple – version 2)</strong></td>
<td>$\min_{i=1} \left( ss(b_{ij}) \right)<em>{i} n</em>{r}$</td>
<td>$\min_{i=1} \left( ad(b_{ij}) \right)<em>{i} n</em>{r}$</td>
<td></td>
</tr>
<tr>
<td><strong>Column-dominant</strong></td>
<td>$\min_{j=\text{argmax}} \left( \frac{ss(b_{ij})}{i} \right)<em>{j} h</em>{c}$</td>
<td>$\min_{j=\text{argmax}} \left( \text{Me}(b_{ij}) - \text{Me}(b_{jk}) \right)<em>{j} a</em>{d}(b_{ij}) h_{c}$</td>
<td></td>
</tr>
<tr>
<td><strong>(Simple – version 2)</strong></td>
<td>$\min_{i=1} \left( ss(b_{ij}) \right)<em>{i} n</em>{r}$</td>
<td>$\min_{i=1} \left( ad(b_{ij}) \right)<em>{i} n</em>{r}$</td>
<td></td>
</tr>
</tbody>
</table>

---

64
<table>
<thead>
<tr>
<th>Block inconsistencies - $\delta(R(C_a, C_b), T)$</th>
<th>Position of the block</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Ideal block}$</td>
<td>sum of squares</td>
</tr>
<tr>
<td>$\min_{i=\arg \max {B_{i,j}}} \left( \sum_{j=1}^{n} \left( \sum_{i:j \neq \arg \max {B_{i,j}}} \left( \sum_{j} \left( b_{ij} - \sum \text{diag}(B) \right) \right)^2 \right) \right)_{n_r}$</td>
<td>$\min_{i=\arg \max {M_{i,j}}} \left( \sum_{j=1}^{n} \left( \sum_{i:j \neq \arg \max {M_{i,j}}} \left( \sum_{j} \left( b_{ij} - \sum \text{Me(diag}(B) \right) \right)^2 \right) \right)_{n_r}$</td>
</tr>
<tr>
<td>$\min_{j=\arg \max {B_{i,j}}} \left( \sum_{i=1}^{n} \left( \sum_{j} \left( b_{ij} - \sum \text{diag}(B) \right) \right)^2 \right)_{n_r}$</td>
<td>$\min_{j=\arg \max {M_{i,j}}} \left( \sum_{i=1}^{n} \left( \sum_{j} \left( b_{ij} - \sum \text{Me(diag}(B) \right) \right)^2 \right)_{n_r}$</td>
</tr>
<tr>
<td>row-functional</td>
<td>$\sum_{i=1}^{n} \left( \sum_{j=1, j \neq \arg \max {B_{i,j}}} b_{ij}^2 \right)$</td>
</tr>
<tr>
<td>column-functional</td>
<td>$\sum_{j=1}^{n} \left( \sum_{i=1, i \neq \arg \max {B_{i,j}}} b_{ij}^2 \right)$</td>
</tr>
</tbody>
</table>
### Legend:

- The null block type is included in this list mostly for continuity reasons. Unless the conditions in brackets marked with * in Table 7.3 are used, it is not necessary as it is generally only a special case of all other block types. If the conditions marked with * are used, the inclusion of a null block type is necessary. The null block inconsistencies presented in this table are formulated based on the inconsistencies for a complete block. In some cases, other ways of computing null block inconsistencies may be desired.

<table>
<thead>
<tr>
<th>Ideal block</th>
<th>Block inconsistencies - $\delta(R(C_a,C_b),T)$</th>
<th>Position of the block</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum of squares</td>
<td>absolute deviations</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>row-f-regular</th>
<th>$ss(f(B_{[i]}))n_c$</th>
<th>$ad(f(B_{[i]}))n_c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>column-f-regular</td>
<td>$ss(f(B_{[j]}))n_r$</td>
<td>$ad(f(B_{[j]}))n_r$</td>
</tr>
<tr>
<td>$f$-regular</td>
<td>$\max\left( \frac{ss(f(B_{[i]}))n_c}{n_r}, \frac{ss(f(B_{[j]}))n_r}{n_c} \right)$</td>
<td>$\max\left( \frac{ad(f(B_{[i]}))n_c}{n_r}, \frac{ad(f(B_{[j]}))n_r}{n_c} \right)$</td>
</tr>
</tbody>
</table>

#### B Matrix:

- $B$ matrix of block $R(C_a,C_b)$
- $B_{[i]}$ the $i$-th row of the matrix $B$
- $B_{[j]}$ the $j$-th column of the matrix $B$
- $b_{i,j}$ an element of matrix $B$ defined by $i$-th row and $j$-th column
- $n_c$ number of rows in a block = $\text{card} C_i$
- $n_r$ number of columns in a block = $\text{card} C_j$

#### diag($B$):

A vector of the diagonal elements of the matrix $B$

#### $Me(x)$:

Median

#### $\bar{x}$:

Arithmetic mean

#### $ss(x)$:

Sum of square deviations from the mean: $ss(x) = \sum (x_i - \bar{x})^2$

#### $ad(x)$:

Sum of absolute deviations from the median: $ad(x) = \sum |x_i - Me(x)|$
The block type inconsistencies can also be adjusted for pre-specified blockmodeling in the sense of values by substituting the mean ($X$) or median ($\text{Me}(x)$) (as a value from which deviations are computed) with a pre-specified value. The block inconsistency for the null block can be seen as an example of how the complete block inconsistency can be adjusted for a pre-specified value 0.

The block type inconsistencies for most block types presented in Table 7.4 follow quite naturally from the descriptions of their ideal blocks in Table 7.3. The only exception is the block inconsistency for an \(f\)-regular block.

### 7.2.1 \(f\)-regular block inconsistency

Several possibilities were considered for the computation of \(f\)-regular block inconsistency. This inconsistency should be computed taking both row-\(f\)-regular block inconsistency and column-\(f\)-regular block inconsistency into account. Other possibilities have been considered for combining these two block type inconsistencies, such as using the functions sum and mean instead of the maximum. Although the use of the sum might seem the most logical, this might make the \(f\)-regular block inconsistency too large. A similar approach was taken in valued blockmodeling, however special care was taken there in order to make sure that each cell can contribute only once to the \(f\)-regular block inconsistency. This approach is not possible here.

The maximum was eventually chosen since it preserves the inequalities of block inconsistencies that hold for both binary and valued blockmodeling:

$$\delta(R(C_a,C_b),\text{cre}) \leq \delta(R(C_a,C_b),\text{reg})$$

and

$$\delta(R(C_a,C_b),\text{rre}) \leq \delta(R(C_a,C_b),\text{reg}).$$

Row(column)-\(f\)-regular and complete block types are compatible in the sense that the block inconsistencies for homogeneity complete block and homogeneity row(column)-\(f\)-regular block match if and only if the rows(columns) are homogeneous, that is if each row (column) has zero variance. However, the inequalities

$$\delta(R(C_a,C_b),\text{cre}) \leq \delta(R(C_a,C_b),\text{reg}) \leq \delta(R(C_a,C_b),\text{com})$$

and

$$\delta(R(C_a,C_b),\text{rre}) \leq \delta(R(C_a,C_b),\text{reg}) \leq \delta(R(C_a,C_b),\text{com})$$

hold in sum of squares blockmodeling if function \(f\) is mean, but not generally.

Therefore, it is questionable if \(f\)-regular and similar\(^{57}\) block types are compatible with the other block types when function \(f\) is not the mean or the measure of deviation is not the sum of squared deviations. They are compatible in the sense that e.g. row-\(f\)-regular block inconsistency is lower than complete block inconsistency if the variation (an appropriate normalized version that takes into account the number of units, e.g. variance for sum of

---

\(^{57}\) row-\(f\)-regular and column-\(f\)-regular
squares blockmodeling) in values of function $f$ computed on the rows is smaller than the variation in all cell values. However, for some functions $f$, e.g. the maximum which is also the function most frequently used as $f$, the variation in values of function $f$ computed on the rows is often greater than the variation of all cell values in the block. Therefore, at least in general, the above inequalities do not hold. Also, depending on the function $f$ used the block type inconsistencies for $f$-regular and similar blocks can become very large. If this is not seen as problematic, $f$-regular and similar blocks can be declared compatible with other block types and therefore used together.

If they are to be used together then caution is advisable. This is a problem since, as was seen in Subsection 3.3.3, the maximum is suggested as an appropriate function. Also, the results of the simulations presented in Chapter 10 show that using the mean as function $f$ usually leads to considerably worse results than those obtained using the maximum as function $f$. In cases where row-max-regular, column-max-regular and/or max-regular block types are used in the same blockmodel such as e.g. a complete block type, other approaches might be more appropriate. This is a big disadvantage for homogeneity blockmodeling since one of the main ideas and advantages of generalized blockmodeling is to combine several types of blocks in the same blockmodel and $f$-regular and similar blocks are some of the most important block types.

7.2.2 Short comparison with previously introduced approaches

The main advantage of these approaches over those presented in the previous section is that they do not require an extra parameter (in the other approach determining the correct value of $m$ posed considerable problems). Therefore, if nothing else, the solutions obtained by this blockmodeling type could be used as a stepping stone for the previously described valued blockmodeling. They can also be used with negative values on the ties, although for some block types (e.g. row- and column-dominant version 3 and row- and column-functional), additional subtypes might be needed, where the maximum would be replaced by the minimum.

A similar approach at least for structural equivalence was already suggested by Borgatti and Everett (1992b: 101), where they stated that the average variance within matrix blocks is a natural measure of fit. The disadvantage of this measure compared with the sum of squares and absolute deviations measures is that, in the variance case, the size of the blocks has a large effect on the contribution of an individual cell to the inconsistency.

The generalized blockmodeling approach presented by Doreian et al. (2005: 226) suggests that block inconsistencies can be normalized by dividing them\(^{58}\) by the number of cells in a block. This is also possible in approaches to generalized blockmodeling for valued networks. In this case, the sum of squares approach changes into the variance approach.

---

\(^{58}\) They actually divide the non-normalized block inconsistency + 1 \(\delta(R(C_a, C_b), T) + 1\) by the number of cells in a block (Doreian et al., 2005: 226).
### 7.3 Implicit blockmodeling

This section is based on the work of Batagelj and Ferligoj (2000: 11-13) presented in Section 6.4.

The descriptions of ideal blocks for implicit blockmodeling are very similar as those for homogeneity blockmodeling presented in Table 7.3. However, there are slight differences. Therefore, the descriptions of ideal blocks for implicit blockmodeling are presented in Table 7.5.

#### Table 7.5: Descriptions of ideal blocks for implicit blockmodeling

<table>
<thead>
<tr>
<th>Ideal block with ‘label’</th>
<th>Description for implicit blockmodeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null* ‘null’</td>
<td>All 0 (except may be diagonal, where all are then equal)</td>
</tr>
<tr>
<td>Complete ‘com’</td>
<td>All equal and non-zero* (diagonal may be treated separately)</td>
</tr>
<tr>
<td>Row-dominant ‘rdo’</td>
<td>there exists a row where all values are equal and non-zero* (except may be diagonal, where all are then equal)</td>
</tr>
<tr>
<td>Col-dominant ‘cdo’</td>
<td>there exists a column where all values are equal and non-zero* (except may be diagonal, where all are then equal)</td>
</tr>
<tr>
<td>Row-functional ‘rfn’</td>
<td>maximums (must be unique) of all rows are equal and non-zero*, all other values 0</td>
</tr>
<tr>
<td>Column-functional ‘cfn’</td>
<td>maximums (must be unique) of all columns are equal and non-zero*, all other values 0</td>
</tr>
<tr>
<td>Row(-max)-regular ‘rre’</td>
<td>function $\max$ on each row is equal and non-zero* for all rows</td>
</tr>
<tr>
<td>Column(-max)-regular ‘cre’</td>
<td>function $\max$ on each column is equal and non-zero* for all columns</td>
</tr>
<tr>
<td>(max-)regular ‘reg’</td>
<td>function $\max$ on each row and each column is equal and non-zero* for all rows and all columns separately</td>
</tr>
</tbody>
</table>

Legend:  
* - The condition that the values should be non-zero is optional and depends on the version of implicit blockmodeling used. See Subsections 7.3.1 and 7.3.2 and Section 7.4.

The original formulas of Batagelj and Ferligoj (2000: 13) for three block type inconsistencies are presented in Table 6.4. These block type inconsistencies are normalized to take a value between 0 and 1.

To make their approach more similar to the other approaches presented in this chapter, their values should be multiplied by the number of cells in a block, as in the original formulas, the
block type inconsistencies were effectively divided by the number of cells in the block. This kind of normalization (dividing the block type inconsistency by the number of cells) is later referred to as block size normalization. It can also be seen that the formulas presented in Section 6.4 are for off-diagonal blocks.

On the basis of these examples, binary blockmodeling and the proposed approaches presented in this chapter, the block type inconsistencies for the remaining block types (without normalization using block size) were developed. They are also rewritten in a way that the similarity with valued blockmodeling becomes more obvious. They are presented in Table 7.6. All of these formulas include the division of the numerator by the maximum of all non-zero numbers of either a block or just one row or column of a block. Regarding the question of how this is defined when there are no non-zero values in a block or selected row or column, see Subsection 7.3.1.

Table 7.6: Computation of block type inconsistencies for implicit blockmodeling without block size normalization

<table>
<thead>
<tr>
<th>Block type</th>
<th>block type consistencies - $\delta(R(C_j, C_j), T)$</th>
<th>Position of the block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>$\sum B / \max {B \neq 0}$</td>
<td>off-diagonal</td>
</tr>
<tr>
<td></td>
<td>$= \sum B + \min(0, -2 \sum \text{diag}(B) + \max {B \neq 0} n_r) / \max {B \neq 0}$</td>
<td>diagonal</td>
</tr>
<tr>
<td></td>
<td>$= \sum B + \min(0, - \sum \text{diag}(B) + \sum (\max {B \neq 0} - \text{diag}(B))) / \max {B \neq 0}$</td>
<td>diagonal</td>
</tr>
<tr>
<td>Complete</td>
<td>$n_r n_r - \sum B / \max {B \neq 0} = \sum (\max {B \neq 0} - B) / \max {B \neq 0}$</td>
<td>off-diagonal</td>
</tr>
<tr>
<td></td>
<td>$= \sum (\max {B \neq 0} - B) + \min(0, 2 \sum \text{diag}(B) - \max {B \neq 0} n_r) / \max {B \neq 0}$</td>
<td>diagonal</td>
</tr>
<tr>
<td></td>
<td>$= \sum (\max {B \neq 0} - B) + \min(0, \sum \text{diag}(B) - \sum (\max {B \neq 0} - \text{diag}(B))) / \max {B \neq 0}$</td>
<td>diagonal</td>
</tr>
<tr>
<td>Row-dominant</td>
<td>$n_r \sum \max {B[j,i] \neq 0} / \max {B[j,i] \neq 0} = \min_i \left{ \sum_j \left( \max_j {B[j,i] \neq 0} - B[j,i] \right) / \max_j {B[j,i] \neq 0} \right} n_r$</td>
<td>off-diagonal</td>
</tr>
<tr>
<td></td>
<td>$= \sum \left( \max_j {B[j,i] \neq 0} - B[j,i] \right) - \min_i \left( \max_j {B[j,i] \neq 0} - 2B[j,i] \right) / \max {B[j,i] \neq 0} n_r$</td>
<td>diagonal</td>
</tr>
</tbody>
</table>

70
By comparing the formulas in Table 7.6 for implicit blockmodeling and Table 7.2 for valued blockmodeling, we see they are very similar. There are in fact only two differences.

Parameter $m$ in valued blockmodeling is replaced in implicit blockmodeling by the maximum of either entire block or by the maximum of the appropriate row or column. Due to the fact that this maximum is always greater or at least equal to the value that is deducted from it,
there is no need to take only positive values into account, as is done with valued blockmodeling.

The second difference is that the block type inconsistency is here divided by the maximum (corresponding to parameter $m$ in valued blockmodeling) used in its computation. This normalization is later referred to as *maximum normalization*. This would have no effect on valued blockmodeling since $m$ is always the same. However, with implicit blockmodeling the maximum is different across blocks and sometimes even within one block across different block types.

To make these formulas even more comparable with valued blockmodeling, we could omit the division with the maximum of the appropriate values (maximum normalization). The formulas obtained in this way are presented in Table 7.7.

### 7.3.1 Options for determining the maximum of non-zero values when all values in a block (or appropriate row or column) are 0

As can be seen from Table 7.6 and Table 7.7, the maximum of non-zero values in a block (or appropriate row or column in row- or column-dominant blocks) plays an important role in computing block inconsistencies for implicit blockmodeling. The question arises of what should be done when there are no non-null values, which means that all values are 0.

In the original version of implicit blockmodeling (Batagelj and Ferligoj, 2000: 12-13) the problem was solved by making the fraction where such a maximum occurs 0, thus making in the case of only 0s in block inconsistencies for null block 0 and 1 for all other block types.

However, this is not a satisfactory solution when the block inconsistencies are not computed in the normalized form since we need an actual value for the maximum. Two options are considered here. The maximum of non-zero elements is, when all elements are 0, set to:

1. the maximum of a larger structure than that for which the maximum is computed. In case maximum or non-zero elements should be computed for a block, the maximum value of the whole matrix (network) is used instead. If it should be computed for a row a maximum of a block is used instead, and if that one is also 0 the maximal value of the whole matrix (network) is used.

2. 0, and if this is normalized, the inconsistency for any block (or row or column) is set to 0.

It is also possible to select one of these options when computing the maximums of non-null elements of blocks and the other when computing the maximums of non-null elements of a single row or column in a block.

The first option provides in the normalized case the same results as the treatment suggested by Batagelj and Ferligoj (2000: 12-13). Actually, any positive number would produce the same results in the normalized case. However, this solution also provides reasonable block type inconsistencies for a non-normalized version.
Table 7.7: Computation of block type inconsistencies for implicit blockmodeling without maximum and block size normalization

<table>
<thead>
<tr>
<th>Block type</th>
<th>Block inconsistencies - $\delta(R(C_i, C_j), T)$</th>
<th>Position of the block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>$\sum B$</td>
<td>off-diagonal</td>
</tr>
<tr>
<td></td>
<td>$\sum B + \min(0, -2\sum \text{diag}(B) + \max{B \neq 0} n_r)$</td>
<td>diagonal</td>
</tr>
<tr>
<td></td>
<td>$\sum B + \min(0, -\sum \text{diag}(B) + \sum(\max{B \neq 0} - \text{diag}(B)))$</td>
<td></td>
</tr>
<tr>
<td>Complete</td>
<td>$\sum (\max{B \neq 0} - B)$</td>
<td>off-diagonal</td>
</tr>
<tr>
<td></td>
<td>$\sum (\max{B \neq 0} - B) + \min(0, 2\sum \text{diag}(B) - \max{B \neq 0} n_r)$</td>
<td>diagonal</td>
</tr>
<tr>
<td></td>
<td>$= \sum (\max{B \neq 0} - B) + \min(0, \sum \text{diag}(B) - \sum(\max{B \neq 0} - \text{diag}(B)))$</td>
<td></td>
</tr>
<tr>
<td>Row-dominant</td>
<td>$\min_i \left( \sum_j \left( \max{B_{i,j} \neq 0} - B_{i,j} \right) \right) n_c$</td>
<td>off-diagonal</td>
</tr>
<tr>
<td></td>
<td>$\min_i \left( \sum_j \left( \max{B_{i,j} \neq 0} - B_{i,j} \right) - \min\left(0, \max{B_{i,j} \neq 0} - 2B_{i,j}\right) \right) n_r$</td>
<td>diagonal</td>
</tr>
<tr>
<td>Column-dominant</td>
<td>$\min_j \left( \sum_i \left( \max{B_{i,j} \neq 0} - B_{i,j} \right) \right) n_c$</td>
<td>off-diagonal</td>
</tr>
<tr>
<td></td>
<td>$\min_j \left( \sum_i \left( \max{B_{i,j} \neq 0} - B_{i,j} \right) - \min\left(0, \max{B_{i,j} \neq 0} - 2B_{i,j}\right) \right) n_r$</td>
<td>diagonal</td>
</tr>
<tr>
<td>Row-max-regular</td>
<td>$\sum_i \left( \max{B \neq 0} - \max{B_{i,i}} \right) n_c$</td>
<td></td>
</tr>
<tr>
<td>Column-max-regular</td>
<td>$\sum_j \left( \max{B \neq 0} - \max{B_{j,j}} \right) n_r$</td>
<td></td>
</tr>
<tr>
<td>Max-regular</td>
<td>$\sum_{i=1}^{n_i} \sum_{j=1}^{n_j} \max{\max{B \neq 0} - \max{B_{i,k}}, \max{B \neq 0} - \max{B_{i,j}} }$</td>
<td></td>
</tr>
<tr>
<td>Row-functional</td>
<td>$\sum_{i=1}^{n_i} \left( \max{B \neq 0} - \max{B_{i,i}} \right) n_c + \sum_{j=1, j=\arg\max(B_{i,j})}^{n_j} B_{i,j}$</td>
<td></td>
</tr>
<tr>
<td>Column-functional</td>
<td>$\sum_{j=1}^{n_j} \left( \max{B \neq 0} - \max{B_{i,j}} \right) n_r + \sum_{i=1, i=\arg\max(B_{i,j})}^{n_i} B_{i,j}$</td>
<td></td>
</tr>
</tbody>
</table>
The second solution is practically the opposite. Although it might seem strange that in the
case of an ideal null block the inconsistency for a complete block would also be 0, the need
for such an option arises when we do not use null blocks. The use of this second option is
actually only advisable in cases when the null block type is not one of the allowed block types
in any block of the blockmodel. The second option allows that the null block type is a special
case of all other block types, similarly as it is possible with homogeneity blockmodeling.
However, as this option is natural for homogeneity blockmodeling it is foreign to implicit
blockmodeling.

7.3.2 Different versions of implicit blockmodeling

Implicit blockmodeling mainly has three options that can be selected and which result in
different versions of implicit blockmodeling. These options are whether or not:
1. the null block type should be allowed when not using pre-specified blockmodeling;
2. maximum normalization should be used; and
3. block size normalization should be used.

Although it would almost always be desired to allow null blocks in the model, as it is
certainly important to distinguish between null and other block types, initial experience with
the use of implicit blockmodeling suggests that this may lead to unsatisfactory solutions. The
reason is that adding a relatively59 large value to any block makes its inconsistency with all
block types except the null block type very large. If a maximum normalization is also used,
this makes the null block inconsistency actually even smaller. Even if a maximum
normalization is not used, the null block inconsistency becomes relatively smaller compared
to block type inconsistencies of other block types, as they become larger. This can make a
block misclassified, as is also shown in the next section. In addition, it of course also affects
which partition is optimal.

When applying implicit blockmodeling to empirical and simulated networks, using implicit
blockmodeling without the null block type as one of the allowed block types usually produced
better partitions than with the null block type. An exemption is where pre-specified
blockmodeling is used. As might be expected, this problem is greater when the distribution of
tie (or cell) values has a long right tail. On one hand, the use of the null block type as one of
allowed block types is therefore not advised. As the example in the next subsection indicates,
this might however lead to finding inappropriate partitions. On the other hand, it should be
pointed out that, although the removal of the null block type from the allowed block types
may produce better partitions, this disrupts the logic of implicit blockmodeling. Therefore, it
is questionable if such an approach is sensible.

The use of a maximum normalization is especially not advised when the null block type is
used due to the effects presented above. If the null block type is not used, the question

59 Relative to other values in the block.
changes primarily to whether the deviation of a unit size is more serious when it is a deviation from a relatively low value as compared to a deviation of a unit size from a relatively high value. While initial experience advises against the use of maximum normalization, this experience is limited to only a few cases. However, it is also supported by some simulation results presented in Chapter 10.

The issue of using block size normalization is the same as with other blockmodeling types. In principle, its usage is not advised as it means that block size influences the contribution of an individual cell to the total inconsistency.

### 7.3.3 Comparison with valued blockmodeling and the problems with classification

As mentioned, the formulas for both implicit and valued blockmodeling are very similar. The two approaches support practically all block types introduced by binary blockmodeling. However, the two differences between valued and implicit blockmodeling described above have a large effect on the performance of these blockmodeling types, especially due to the fact that the maximum changes between blocks and even within blocks for some block types.

This comparison is made using implicit blockmodeling with a maximum normalization and without using block size normalization, that is using the block type inconsistencies presented in Table 7.6. However, the results are also the same if the original approach (with block size normalization) is used or if maximum normalization is not used.

The substitution of parameter \( m \) with the maximum of either entire block or only an appropriate row or column brings its own advantages and disadvantages. The main advantage is that, similarly to homogeneity blockmodeling, no parameter has to be specified in advance. Therefore, implicit blockmodeling combines the advantages of valued (compatible block types, the null block type that is not a special case of other block types) and homogeneity (no parameter needed) blockmodeling. However, the substitution of \( m \) with a maximum value also poses a serious threat. This is clearly seen in the next example. Consider the network and the partition\(^{60}\) presented with the matrix in Figure 7.1.

The problem is with the classification of blocks produced by implicit blockmodeling. We would expect to get the following image (which is produced by both binary blockmodeling and valued blockmodeling with \( m = 1 \)):

```
1   2
1 "null" "com"
2 "com" "null"
```

---

\(^{60}\) The partition in that matrix is the only optimal partition using binary blockmodeling, valued blockmodeling with \( m = 1 \) and homogeneity blockmodeling (both measures of variability) according to structural equivalence. When using implicit blockmodeling, the partition is always optimal when maximum normalization is used (regardless of whether block size normalization and the null block type are used). When the null block type is and block size normalization is not used, the partition in Figure 7.1 is also optimal if maximum normalization is not used.
However, this is not the result we get with implicit blockmodeling. The image we get with implicit blockmodeling is (see Appendix A for more details):

```
1 2
1 "null" "com"
2 "null" "null"
```

*Figure 7.1: A network with an optimal partition*

![Network with optimal partition](image)

Although it could be discussed when a block should be considered null and when complete\(^{61}\), it does not seem right that block \(R(2, 1)\) (lower left) should be null if block \(R(1, 2)\) is complete. The two blocks are almost the same, with the only difference being the value 3 in block \(R(2, 1)\) in place of the value 1 in block \(R(1, 2)\). So block \(R(2, 1)\) has all values equal or greater to those of block \(R(1, 2)\), yet block \(R(1, 2)\) is considered complete and block \(R(2, 1)\) null. This might be problematic since a complete block should indicate stronger ties than a null block.

As mentioned in the previous subsection, this often does not only affect the classification of blocks but also the partition that is optimal under such an approach. Initial experience suggests that when not using pre-specified blockmodeling the null block type should not be one of the allowed block types. However, in the example presented above using implicit blockmodeling without a null block type and without using block size and maximum normalization would lead to a different partition (a ‘wrong’ partition) to that presented in Figure 7.1.

---

\(^{61}\) For valued networks, it is not necessary that for example block \([1, 2]\) is considered complete since it is possible that ties with a value of 1 are not strong enough to be drawn in the reduced graph.
7.4 The null block type problem

The null block problem applies to homogeneity and in some cases also to implicit blockmodeling. Here, it will mainly be discussed from the point of view of homogeneity blockmodeling as the problem is more pressing there. At the end, how and in which cases it also applies to implicit blockmodeling will be additionally mentioned.

7.4.1 Description of the problem and why it occurs

The null block type problem occurs when there are problems with classifying empirical blocks as the null block type or as some other block type. This is usually a serious problem since the distinction between a null block type and any other block type is the most important distinction. An empirical block that is classified as any other block type other than null induces a tie in the reduced graph of the network, while an empirical null block does not. Needless to say, the reduced graph is one of the most important results of generalized blockmodeling.

The problem occurs when the value 0 (in a matrix representing a network) has a special meaning. Namely, it does not represent a tie with value of 0 but means that the tie does not exist. Although this is true to some extent for all blockmodeling types, homogeneity blockmodeling in particular treats the value 0 in a matrix like any other value. This is the core of the problem.

In other blockmodeling types for valued networks, the value 0 by itself does not have any special meaning in terms of the difference between two values in a matrix being greater, just because one of them is 0 (e.g. the difference from 0 to 1 being greater than the difference from 2 to 1). Yet they do put it in a special position when defining the null block type. However, to make use of this special position of the value 0 when classifying blocks as null or as some other block type, another value has to be given as an opposite to 0. In valued blockmodeling, such a value is the parameter \( m \) while in implicit blockmodeling such a value is the maximum of the block or row or column that is in focus at the time. On the other hand, homogeneity blockmodeling does not have any such value.

The problem does not exist when the value 0 in the original network does not have a special meaning. In such cases, the null block type also does not have to have a special meaning. The fact that it is only a special case of the complete block type (and based on the descriptions also of most other block types) is not a problem. With homogeneity blockmodeling, it is essential that each block is also characterized by some value (usually the central values). If 0 does not have a special status then the null block type also does not have a special status. Then it is perfectly all right if a null block type is represented by a complete block type (or is a special case of a compete block type) with a central value of 0 (or very close to it if it not an ideal null block). There is actually no need to differentiate between the null and other block types. The information about the central value of the block is enough. In such cases, null blocks are not even necessary. However, this is rarely the case in real networks.
When the null block type is a special case of most other block types

When the condition marked with a ‘*’ in Table 7.3 is omitted, the problem is that the null block type is a special case of the other block types. This makes the classification of blocks as null or some other block type problematic. Two strategies can be taken, although both can be seen as a cure of symptoms rather than of the original problem:

1. We can decide that an empirical block will be classified as a null block type only if null block inconsistency is lower than or equal to the block type inconsistency of the competing block. For sum of squares blockmodeling, null block type inconsistency can never be lower than complete block inconsistency and can only be equal to it in the case of an ideal null block. In practice, it is also very unusual for it to be lower than the block type inconsistencies of other block types (with which it is compatible). This means that for sum of squares blockmodeling only ideal null blocks are classified as the null block type. For absolute deviations blockmodeling, a block where at least half of the cells have a value of 0 would have null block inconsistency equal to complete block inconsistency. Even compared to other block types, the null block type is not in such a disadvantage as with sum of squares blockmodeling. Therefore, this option is much more suitable for absolute deviations blockmodeling than for sum of squares blockmodeling. This strategy can be accomplished within homogeneity blockmodeling by including the null block type as one of the allowed block types and using its inconsistency as defined in Table 7.4.

2. The second option is that we do not include the null block type as one of the allowed block types and therefore do not use null block inconsistencies within homogeneity blockmodeling. If we still want to obtain a classification of empirical blocks into null and other block types, other means (rules or procedures outside generalized blockmodeling) must be used to determine when a tie is present in a reduced graph. For example, a threshold $t$ may be selected. If the value from which the deviations (the central value) is lower than threshold $t$, then the block is classified as null. Otherwise, it is classified as the block type in which it was classified during homogeneity blockmodeling.

When a null block type is not a special case of most other block types

If the condition marked with a ‘*’ in Table 7.3 is enforced, the problem is as follows. The block type inconsistencies defined in Table 7.4 of block types other than the null are not

---

62 The condition marked with a ‘*’ is present in the descriptions of ideal blocks for most block types other than the null block type. It states all or some values should not be zero.

63 It is questionable if $f$-regular, row-$f$-regular, and column-$f$-regular blocks are compatible with other block types (including the null block type), if function $f$ is not a mean or the measure of deviation is not the sum of squared deviations.

64 The condition marked with a ‘*’ is present in the descriptions of ideal blocks for most block types other than the null block type. It states that either some or all values should not be zero.
compatible with the description of their ideal blocks in Table 7.3. The block type inconsistencies as defined in Table 7.4 are 0 even for empirical blocks that are not ideal blocks of the corresponding block type, but are ideal null blocks. The problem is also that when the empirical block is not an ideal null block but very close to an ideal null block, the null block inconsistency is always higher than e.g. the complete block inconsistency. The null block inconsistency is usually (but not necessarily) also higher than the block types inconsistencies of other block types previously mentioned. This shows that the null block type is somewhat hampered.

The problem can be solved in several ways:

1. Although we acknowledge that the null block is not a special type of other block types (as we define them to explicitly demand that the values that are equal are not zero) we still use the same procedure as suggested in Subsection 7.4.2. Depending on which option from Subsection 7.4.2 we choose, we modify the block type inconsistencies from Table 7.4 so that they are compatible with the description of an ideal block in Table 7.3 in the following ways:

   a. If we select option 1, we have to modify the inconsistencies of other block types (other than the null) by setting their value in the case of an ideal null block to infinity or some other value that is at least as large as the largest possible block inconsistency for the empirical block that is to be classified.

   b. If we select option 2, we have to modify the inconsistencies of other block types (other than the null) by setting their value to infinity or some other value that is at least as large as the largest possible block inconsistency for the empirical block that is to be classified, whenever the central value (from which the deviations are computed) is larger than threshold $t$.

2. The main reason block type inconsistencies cannot be used as they are presented in Table 7.4 is that the null block type is hampered. Therefore, something should be done to make it competitive with the other block types. Two possibilities are suggested here:

   a. A smaller weight $w(T)$ is given to the null block type. This in essence lowers the block type inconsistency. However, it is hard to determine how much smaller this weight should be.

   b. The null block type is hampered because the sum of squared deviations or sum of absolute deviations is not allowed to be calculated from the optimal value (the mean for the sum of squared deviations and the median for the sum of absolute deviations), but is always calculated from 0. Something similar could be done for the other block types to thus put them on more equal grounds. However, restricting all other block types can be somewhat too stringent and may eliminate most of the advantages of homogeneity blockmodeling. It is probably only justified when enough information is available and this
information is incorporated in the sense of pre-specified blockmodeling. When such information is not available, it might still be sensible to set a lower limit $l$ for the value from which the deviation is computed in the block type inconsistencies of these block types, while leaving it otherwise free. This would hamper these block types only when they have to compete with the null block type, which is when they are relatively similar to the ideal null block. This lower limit $l$ is also much easier to set than, for example, the weight of the null block as suggested in paragraph 1 above. However, as far as classification of the block is concerned this is just a fancy way of achieving exactly the same classification as with approach 1.a suggested in this subsection. For the sum of squares approach, the same classification is achieved when either using threshold $t$ in approach 1.a or the lower limit $l = 2t$ using the approach in this paragraph. Of course, using this approach would also affect the partition that would be found. It would probably find partitions that induce blocks which are either nearer to ideal null blocks or further from them.

3. Combine the block type inconsistencies for homogeneity blockmodeling with those for some other blockmodeling. For example, the homogeneity block type inconsistency for a given block type could be summed together with the inconsistency of binary blockmodeling for the same block type. The two inconsistencies should of course be appropriately weighted so that the effects of both blockmodeling types can influence the optimization of the partition and classification of the blocks. The appropriate weights are, however, hard to find.

7.4.4 Discussion

Homogeneity blockmodeling was developed explicitly for valued (measured on interval scale) networks. It was not designed to assign special importance to the value 0. It was conceived on the idea that the inconsistency of a block should be measured by some measure of variability. The use of conditions (form previous subsection) and even the null block type itself is not natural to this approach. The same is in some sense also true of the use of pre-specified blockmodeling with homogeneity blockmodeling. This does not mean that the use of these features (conditions, the null block type, and pre-specified blockmodeling) is forbidden. However, caution and the use of subject-specific knowledge are advised in such cases.

In this dissertation, homogeneity blockmodeling is used exclusively in its natural form (without conditions, the null block type, and pre-specified blockmodeling). As such, it is used with great success in the examples presented in Chapter 9 and with even more success in the simulation presented in Chapter 10. The examples in Chapter 9 were based on real networks where 0 did have the meaning of no tie. In the simulations in Chapter 10, the 0 was the

---

65 In the sense that the measures of variability are no longer computed from the mean (sum of squares) or from the median (absolute deviations).
minimal possible value but otherwise did not have any special meaning. Some simulations were also made where there was an attempt to give the value 0 the special meaning it usually has in networks. In these simulations, the performance of homogeneity blockmodeling was similar to the rest of the simulations.

### 7.4.5 Implication for implicit blockmodeling

As mentioned at the start of this section, the section mainly applies to homogeneity blockmodeling. Implicit blockmodeling in the form suggested by Batagelj and Ferligoj (2000: 12-13) does not suffer from this problem. Also, most of the developments described in Section 7.3 do not introduce this problem.

The problem occurs if we opt not to use the null block type (and null block inconsistency) in the optimization and classification process and to treat the null block type as a special case of all other block types. Such a decision seems counterintuitive, especially if we are aware of the problems caused by the null block type being a special case of most other block types in homogeneity blockmodeling.

However, the idea of such an action was promoted by the fact that when using null block types and null block inconsistencies as introduced in Section 7.3 the classification of empirical blocks into the null or complete block type can be counterintuitive, as shown in Subsection 7.3.3. The empirical examples presented in Chapter 9 and results of simulations presented in Chapter 10 show that not using null block inconsistency (and therefore the null block type) in the optimization process can lead to better partitions.

In such cases, we can then still classify blocks as null or some other block type using some other means or rules, similarly as suggested for homogeneity blockmodeling. For example, we could set a rule that whenever the maximum used in the calculation of the inconsistency is lower than some threshold $t$, we change the classification of that empirical block to the null block type.

Regardless of the fact that the removal of the null block type often improves the performance of the implicit blockmodeling in terms of the partitions found, it is questionable if such a removal is sensible. When null block type is used, implicit blockmodeling has a clear advantage over homogeneity blockmodeling, namely that it includes the null block type that is not a special case of other block types. If the null block type is removed, then this advantage disappears. Implicit blockmodeling becomes very similar to homogeneity blockmodeling. However, implicit blockmodeling without the null block type does not seem theoretically sound. With implicit blockmodeling, the block type inconsistencies for all blocks types other than the null are computed as deviations from a maximal value. Without the null block type as a counterbalance (where inconsistencies are computed from 0), there is little reasoning that would support such a computation of inconsistencies.
7.5 Computing the values of ties in the reduced graph

One advantage of generalized blockmodeling is that, in addition to the partition, it also produces an image, a classification of blocks into different types (null, complete, regular etc.). In the case of a binary network, this is enough to sufficiently describe the types of connections between different clusters in the network. If we want to add more information, we can report block inconsistencies and thus tell the reader how well the ideal blocks fit the empirical block for the block type selected.

However, if we have a valued network it is usually also desired to transfer some information about the values of the ties in the blocks to the image or reduced graph in addition to the block types. This is especially true for homogeneity blockmodeling and implicit blockmodeling when they are used without null blocks since this is then the only way to (systematically) classify some blocks as null.

In the following subsections, different suggestions for computing tie values in a reduced graph are presented. First, the ideas of Batagelj (1997: 148, 151) are presented. This subsection is followed by subsections devoted to different approaches to generalized blockmodeling. In these ‘specialized’ sections, only procedures that are directly linked to the way inconsistencies are computed in a certain generalized blockmodeling approach are presented. The last subsection finishes with some more general suggestions.

7.5.1 Batagelj's suggestions

As mentioned before, binary blockmodeling was also applied to valued networks. Without computing the values of the ties in a reduced graph in some way, all the information about the values of the ties in the original network that is carried over to the reduced one is the value of the slicing threshold $t$. This information only tells us which tie values were actually treated as 1s in the binary blockmodeling and which as 0s. This means that if a block was categorized as complete and not null, it must have more ties with values greater or equal to the threshold $t$ than ties with a lower value (no tie is represented by the value 0). However, this does not tell us if the ties in one complete block are (on average) higher than in another complete block.

To utilize more information about the values of ties when constructing reduced graphs with generalized blockmodeling, Batagelj (1997: 148) presented some ideas for computing the values of the ties in a reduced graph that he called *averaging rules*. To better explain his ideas some additional notation is introduced:

- $V$ is a set of units in the reduced (image) graph (images or representatives of clusters);
- $K \subseteq W \times W$ is the set of connections among these units;
- $v(x,y)$ is a value of a tie from unit $x$ to unit $y$, where $x, y \in U$ or $x, y \in V$; and
- let us denote with $\mu: U \rightarrow V$ a mapping which maps clusters of units to the corresponding units in the reduced graph. Then, we define for $t \in V$

$$C(t) = \mu^{-1}(t) = \{ x \in U : \mu(x) = t \}$$
He suggested the following: For \( t, w \in V \) let \( X = C(t) \) and \( Y = C(w) \). Then the general requirement for an averaging rule \( \bar{\nu} : K \rightarrow S \):

1. \( T(X,Y) = \text{null} \Rightarrow \bar{\nu}(t,w) = 0 \) and
   \[
   \left( \forall p \in R(X,Y) : \nu(p) = c \right) \Rightarrow \bar{\nu}(t,w) = c \text{ or,}
   \]
2. \[
   \sum_{p \in R(X,Y)} \nu(p) = N(t,w)\bar{\nu}(t,w)
   \]
   where \( N(t,w) \) is the *multiplicity* of connection \( (t,w) \). The multiplicity \( N(t,w) \) depends on the type of connections (type of \( T(X,Y) \)) and the ‘dimensions’ of the block. For example:

   \[
   T(X,Y) = \text{com} \Rightarrow N(t,w) = \text{card}(X \times Y)
   \]

   \[
   T(X,Y) = \text{rre} \Rightarrow N(t,w) = \text{card} Y
   \]

   \[
   T(X,Y) = \text{reg} \Rightarrow N(t,w) = \max(\text{card} X, \text{card} Y)
   \]

The above could be stated less formally as follows. The general requirements for averaging rules are:

1. the value of the tie in the reduced graph corresponding to the null block should
   a. be 0 and
   b. if all cells in a block have the value \( c \), then the value of the tie in the reduced block should have the value \( c \) (the same value); or
2. the sum of all cell values in the blocks should equal the value of the tie in the reduced graph multiplied by the *multiplicity* of this tie, which is defined by the dimensions of the block and the type of the block.

These are two competing ideas. While I totally agree with the first one, I do not support the second one. I do not believe this one allows the type of block to sufficiently influence the design of the rule for computing the tie value corresponding to it in the reduced graph.

He also provided examples of averaging rules for interval and ordinal networks. They are presented in Table 7.8.

The examples for the interval scale are made by following the second option for the general requirements presented above. Here with these examples I can more easily explain why I do not support this general requirement. This general requirement and the examples in Table 7.8 and those for the interval scale based on it do not seem right for two reasons:

1. they do not sufficiently take into account the properties of a given block type; and
2. the scaling is inappropriate.

The row-dominant block is a nice example. While the tie value in a reduced graph for a given empirical block would be the average of the block for a complete block, it would be the mean column sum for a row-dominant block. I cannot see a good reason for making the value of the row-dominant block higher than the value of the complete block for the factor of the number of rows. This might be a good idea if the dominant row was truly the only row with ties, but this is not a requirement of the row-dominant block.
Table 7.8: Examples of averaging rules for interval and ordinal networks proposed by Batagelj (1997:151 – Table 4) for block R(X,Y)

<table>
<thead>
<tr>
<th>Ideal block</th>
<th>Interval scale</th>
<th>Ordinal scale</th>
<th>Position of the block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>$v_t/(n_r n_c)$</td>
<td>0</td>
<td>other</td>
</tr>
<tr>
<td>Complete</td>
<td>$v_t/(n_r n_c)$</td>
<td>med $v[X, Y]$</td>
<td>min($s_d, n_r - s_d$)=0</td>
</tr>
<tr>
<td>row-dominant</td>
<td>$v_t/n_c$</td>
<td>med $v[r, Y]$</td>
<td></td>
</tr>
<tr>
<td>column-dominant</td>
<td>$v_t/n_r$</td>
<td>med $v[X, c]$</td>
<td></td>
</tr>
<tr>
<td>row-regular</td>
<td>$v_t/n_r$</td>
<td>med $v[\max X, Y]$</td>
<td></td>
</tr>
<tr>
<td>column-regular</td>
<td>$v_t/n_c$</td>
<td>med $v[X, \max Y]$</td>
<td></td>
</tr>
<tr>
<td>regular</td>
<td>$v_t/\max(n_r n_c)$</td>
<td>min(med $v[\max X, Y]$, med $v[X, \max Y]$)</td>
<td></td>
</tr>
<tr>
<td>row-functional</td>
<td>$v_t/n_r$</td>
<td>med $v[\max X, Y]$</td>
<td></td>
</tr>
<tr>
<td>column-functional</td>
<td>$v_t/n_c$</td>
<td>med $v[X, \max Y]$</td>
<td></td>
</tr>
<tr>
<td>density $\gamma$</td>
<td>$\frac{\gamma v_t/(n_r n_c)}{\gamma v_t/(n_r n_r - n_c)}$</td>
<td>med upper ($\gamma, v[X, Y]$)</td>
<td>other</td>
</tr>
</tbody>
</table>

Legend:

$v_t = \sum_{x \in X, y \in Y} v(x, y)$

$s_d$  diagonal block sum = number of 1s on a diagonal

$n_r$  number of rows in a block = card $X$

$n_c$  number of columns in a block = card $Y$

med  median operator

$r, c$  the dominant row/column

However, this example demonstrates both points. Why should all values in the block contribute to the value of the tie in the reduced graph for a row-dominant block? The basic characteristic of the row-dominant block is that there is a row in the block that is (ideally) ‘complete’, there are no 0s in it. Therefore, a more natural measure for a row-dominant block would be the mean of that row. This would deal with both points mentioned above. Similar objections can be found for most of other block types, with the exception of a complete and potentially (depending on the definition of regular equivalence for valued networks) row- and column-regular blocks.
Different ways of computing the values of ties in a reduced graph are presented in the following subsections.

The suggestions for ordinal data seem more reasonable as they follow the first option regarding the general requirement. However, the formulas are not very clear. I assume that the following expressions should be rewritten as:

- \( \text{med} \left[ \max_{x \in X} X, Y \right] \rightarrow \max \left( \text{med} \left[ x, Y \right] \right) \)
- \( \text{med} \left[ X, \max_{y \in Y} Y \right] \rightarrow \max \left( \text{med} \left[ X, y \right] \right) \)

However, since this dissertation focuses on valued networks measured on at least interval scales they are not discussed here, although some of the logic they follow can be found in the suggestions in the following subsections.

7.5.2 For binary blockmodeling

There is a procedure of computing tie values in a reduced graph that is directly linked to the ways block type inconsistencies are computed. This is to check for an empirical block that is classified as some block type other than the null, which is the highest threshold \( t \) using which this block is still classified as this block type and not as the null. The value should always be 0 for null blocks. This procedure is also appropriate for ordinal data.

7.5.3 For valued blockmodeling

Valued blockmodeling, similarly to binary blockmodeling, classifies ties into null and other allowed block types. Again, similar to binary blockmodeling, the only information these block types mean in terms of values is stored in the value of parameter \( m \).

A similar procedure as that suggested for binary blockmodeling in the previous subsection can also be used for valued blockmodeling. For a given empirical block classified as some block type other than the null, find the value of parameter \( m \) where the block type inconsistency of the block type in question is the same to the null block inconsistency. This value of \( m \) where both block type inconsistencies are the same can be used as the tie value corresponding to that block in the reduced graph. Let us denote this value by \( m_{\text{max}} \). It should, however, be noted that for complete blocks the mean of all tie values would always be equal or greater than \( m_{\text{max}} / 2 \). The mean would equal \( m_{\text{max}} / 2 \) when all tie values are lower than or equal to \( m_{\text{max}} \), and higher otherwise.

This also means that if all values in a block are \( c \), the value of \( m_{\text{max}} \) would be \( 2c \). Therefore, if we wish to follow the first set of general requirements suggested by Batagelj (1997: 148) (and often obtain more reasonable values), the tie value in the reduced graph should be set to \( m_{\text{max}} / 2 \).

The other option that also takes some of the nature of valued blockmodeling into account is to use the approach suggested for sum of squares blockmodeling (in the next subsection), modified in the following way to be more adapted to valued blockmodeling:
1. For all block types except the null type (this exception can be omitted if the network was subjected to censoring at the value of \( m \) when submitted to valued blockmodeling), censor the tie values in the original network at the value of \( m \).

2. Apply the approach suggested for the sum of squares approach on this censored network, modified further in the cases of:
   - row- and column-\( f \)-regular blocks by censoring the values of function \( f^6 \) over rows or columns at the value of \( m \); or
   - \( f \)-regular blocks by using the following formula:
     \[
     \sum_{i=1}^{n_i} \sum_{j=1}^{n_j} \min\left(\min\left(m, f(B_{ij})\right), \min\left(m, f(B_{ij}^c)\right)\right) / n_i n_j
     \]

### 7.5.4 For homogeneity blockmodeling

For homogeneity blockmodeling, one appropriate value is simply the value from which the sums of squared or absolute deviations are computed. This solves the problem for most block types.

However, for \( f \)-regular blocks there are two such values, the row-\( f \)-regular block inconsistency and the column-\( f \)-regular block inconsistency. The \( f \)-regular block inconsistency is then the maximum of these inconsistencies. At the end, we want only one value per block. One option, to strictly follow the basic concept, would be that we select the value which corresponds to the dimension (row or columns) that had the higher inconsistency. However, I believe a better alternative is to use the mean of these two values.

This is, however, not a problem when the function \( f \) is the mean in sum of squares blockmodeling since in such a case the values for the row and columns are the same and actually match the mean of the block. In Table 7.9 rules for computing tie values in a reduced graph based on homogeneity blockmodeling are presented. These rules are also later suggested as more general rules that are not limited to being used with just homogeneity blockmodeling, especially the version for sum of squares blockmodeling.

#### Table 7.9: Rules for computing tie values in a reduced graph based on homogeneity blockmodeling – general rules

<table>
<thead>
<tr>
<th>Block type</th>
<th>Tie values in the reduced graph</th>
<th>Position of the block</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>complete</td>
<td>Cent B</td>
<td>off-diagonal</td>
</tr>
</tbody>
</table>

\( ^{66} \) Only necessary if function \( f \) does not possess the property \( f(a) \geq \max(a) \), and \( a \) is a valued vector.
### Legend:

- **B** matrix of block \( R(C_i, C_j) \)
- **B\(_{[i]}\)** the i-th row of the matrix \( B \)
- **B\(_{[j]}\)** the j-th column of the matrix \( B \)
- **b\(_{ij}\)** an element of matrix \( B \) defined by i-th row and j-th column
- **nr** number of rows in a block = \( \text{card} \ C_i \)
- **nc** number of columns in a block = \( \text{card} \ C_j \)
- **diag(B)** a vector of the diagonal elements of the matrix \( B \)
- **Cent** a measure of central tendency – mean for sum of squares blockmodeling, median for absolute deviations blockmodeling
- **Cent\(_x\)** dominant row, column – depends on the version of the row-dominant block. See Appendix B

### 7.5.5 For implicit blockmodeling

Similarly as for homogeneity blockmodeling, there is one option that is somehow suggested by the method and that is to employ the maximum that is used in the computation of inconsistencies. The obvious exception is the null block type, where the appropriate value is 0. This maximum is usually the block maximum, with the exception of row and column-dominant blocks where it is the maximum of the dominant row or column. However, this ‘rule’ cannot be used when the null block type is not one of the allowed block types. Even if they are allowed, this rule leads to similar problems as from using the null block type itself, namely that a block with relatively high values and one extremely high value is assigned a value of 0 (since it is classified as a null block type), while a block with essentially the same values, only without the extreme value (it can be even with a 0 in its place) is assigned a relatively high value.

When a null block type is not one of the allowed block types, other rules must be applied. One option is the procedure suggested for the sum of squares approach, modified in a similar way...
as for the valued blockmodeling approach by using the following formula for \((\text{max})\text{-regular}\) blocks:

\[
\sum_{i=1}^{n_i} \sum_{j=1}^{n_j} \min\left(\max B_{[i]}, \max B_{[j]}\right) / n_i n_j
\]

### 7.5.6 General rules

In the previous subsections, rules are derived from the ways inconsistencies were computed in individual blockmodeling types. However, there were no suggestions given for other approaches, e.g. indirect approaches. Further, some of the suggested procedures give values that really have to be interpreted with the characteristics of the method used to obtain them in mind. For a given blockmodel (network, partition and, even the classification of blocks), different procedures (e.g. those suggested in previous subsections) can yield very different results. Therefore, some general rules would be useful in order to compare reduced graphs of partitions blockmodels obtained by using different approaches. However, when using rules that were not designed specifically with the approach in question in mind we have to be aware that different approaches can use slightly different definitions (descriptions) of ideal blocks (and therefore of block types). Therefore, reduced graphs obtained in such a way might not be the optimal for representing the results certain approaches. This must be sacrificed in order to make them comparable.

If we review the suggestions made in this section, we see that especially one rule was used more than the others (although sometimes with a modification). This is the rule suggested by the sum of squares blockmodeling. This is no surprise as its foundation is the use of the mean as an appropriate representative value, computed over values selected/computed by taking the characteristics of the block type into consideration. Therefore, the rule suggested by the sum of squares blockmodeling (Table 7.9) modified in a way as suggested in Subsections 7.5.3 and 7.5.5 (the modification for the regular blocks). The modification is to use the following formula for computing the tie value for \(f\)-regular blocks:

\[
\sum_{i=1}^{n_i} \sum_{j=1}^{n_j} \min\left(f(B_{[i]}), f(B_{[j]})\right) / n_i n_j
\]

### 7.6 Possible improvements/alternatives/uses

#### 7.6.1 Combining criteria functions of several blockmodeling types

Sometimes it might be useful to optimize the partition based on criteria functions considering several blockmodeling types. For example, it might be useful to optimize a partition based on the pattern of ties (binary blockmodeling) and the values of ties (one of the blockmodeling types for blockmodeling valued networks). The criteria functions based on different blockmodeling types could be joined so that the block type inconsistencies based on different blockmodeling types for each block type are weighted and summed. In the case of combining binary blockmodeling and valued blockmodeling, this is appropriate when a change of size \(d\) is more important, if it is from 0 (no tie) to \(d\) (or from \(d\) to 0), that if it is from an arbitrary non-zero value \(x\) to \(x \pm d\).
In this case, it is important to combine the two or more criteria functions or measures of inconsistencies within each block and for each block type separately (block type inconsistencies and not block inconsistencies must be combined). The result should be a new block type inconsistency for each block type, which would be a weighted sum of the original block type inconsistencies. However, either the ‘original’ block type inconsistencies must be somehow normalized or an appropriate weighting scheme must be used to ensure that each blockmodeling type used is given the appropriate weight in the combined criterion function (new block type inconsistency).

For example, the new null block inconsistency obtained by combining binary and valued blockmodeling is presented in Table 7.10 (the notation used is the same as in Table 7.2). It is very important here that we treat the diagonal (in diagonal blocks) the same when computing the inconsistency for binary and valued parts.

Table 7.10: New null and complete block inconsistencies obtained by combining binary and valued blockmodeling

<table>
<thead>
<tr>
<th>Block type</th>
<th>New block type inconsistency</th>
<th>Position of the block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td>$w_1 l(s_i) + w_2 \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij}$</td>
<td>off-diagonal</td>
</tr>
<tr>
<td></td>
<td>$+ \min\left( w_1 (n_x - 2s_{a}) + w_2 \left( \sum (m - \text{diag}(B))^+ - \sum \text{diag}(B) \right) 0 \right)$</td>
<td>diagonal</td>
</tr>
<tr>
<td>complete</td>
<td>$w_1 l(n_x, n_x - s_i) + w_2 \sum_{i=1}^{n} \sum_{j=1}^{n} (m-b_{ij})^+$</td>
<td>off-diagonal</td>
</tr>
<tr>
<td></td>
<td>$+ \min\left( w_1 (-n_x + 2s_{a}) + w_2 (- \sum (m - \text{diag}(B))^+ + \sum \text{diag}(B)) 0 \right)$</td>
<td>diagonal</td>
</tr>
</tbody>
</table>

Legend:

- $w_1$ weight for the binary part of the inconsistency
- $w_2$ weight for the valued part of the inconsistency

In the case of combining binary and valued blockmodeling, it is generally problematic to choose the weights in a way that achieves the desired importance of both approaches. If we want both parts to have approximately equal importance, we could give the binary part a weight of 1 and the valued part a weight of $1/\text{mean of non-null values}$. Then, if all blocks are declared to be null, the inconsistency for binary part equals the inconsistency for the valued part. On the other hand, if we want the existence/nonexistence of a tie to have the same
weight as \( x \) times the effect of a unit change in the tie value, then the appropriate weights would be 1 for the valued part and \( x \) for the binary part.

The problem of giving equal importance to both parts is easily solved if the network is censored (see the next subsection) at the value of parameter \( m \). In such a case, an appropriate weighting is simply the weight 1 for the binary part and \( 1/m \) for the valued part. In this case, the theoretical maximum block type inconsistencies are the same for both blockmodeling types.

Similarly, it is easy to combine binary and implicit blockmodeling (with null blocks, the version where maximum normalization is used), as the inconsistencies of implicit blockmodeling are in this case already normalized to be comparable with binary blockmodeling. Giving both parts an equal weight results in their equal theoretical maximum block type inconsistencies for both blockmodeling types. This might also be a way of reducing the problems of the misclassification of blocks in implicit blockmodeling.

Combining any homogeneity approach with any other approach will probably be very problematic. However, combining homogeneity and binary blockmodeling might be a way of tackling the null block problem described in Section 7.4.

The suggestions for weighting schemas in this subsection were aimed to achieve the equal importance (in one sense or another) of two blockmodeling types for the final criterion function. However, I do not claim that this is desired. The researcher should decide what kind of importance (s)he wants to give to an individual blockmodeling type. If we know the appropriate weighting to achieve the equal importance of different blockmodeling types, so-called neutral weights, it is easy to make a selected approach twice as important as the other approach(es) by multiplying its neutral weight by 2.

### 7.6.2 Censoring the network at a certain value in valued blockmodeling

One problematic aspect of the valued blockmodeling approach is that it seems that the null block type might be in a worse position than, for example, the complete block type. The contribution of each cell in a block to null block inconsistency is not limited, while cell can contribute maximally \( m \) (the value of the parameter) to the complete block inconsistency. This situation could be resolved by censoring the network at a certain value. If this value were at least \( m \), then this would not change the errors where the demand is that something must be at least \( m \), but would only limit the error contribution of those cells that should be 0 to this censoring value. In most cases, the most appropriate value for censoring would probably be \( m \). Lower values than \( m \) are not suggested since they would mean that complete block inconsistency could not equal 0 even theoretically.

If \( m \) and this threshold were equal to the minimum value of the network, the result would be binary blockmodeling (where all ties, regardless of their value, would be treated as ties).
7.6.3 Adaptation to signed networks

Signed networks are networks where tie values can be positive or negative (e.g. like vs. dislike). Binary, valued and implicit blockmodeling as described in this dissertation cannot handle negative tie values. However, Doreian and Mrvar (1996) proposed a partitioning approach to structural balance that can be incorporated in these blockmodeling types. Doreian et al. (2006: 295-325) show that it can be incorporated into binary blockmodeling by means of negative and positive blocks. Negative blocks are blocks where all tie values are non-positive while positive blocks are blocks where all tie values are non-negative. The corresponding block type inconsistencies are defined the same as the positive (for a negative block) and negative (for a positive block) part of the criterion function of Doreian and Mrvar (1996: 153) confined to only one block. In fact, the block type inconsistencies for positive and negative block types are very similar to null block type inconsistency. The only difference is that either only negative ties (for positive blocks) or only positive ties (for negative blocks) contribute to block type inconsistencies. The same can also be done for other blockmodeling types.7

Signed valued networks can also be analyzed outside of structural balance theory. For that, homogeneity blockmodeling can be used as they are since they are not designed explicitly for non-negative networks. For other approaches, new types of blocks should be defined. These could be negative counterparts to the existing blocks, like a negative complete block type, a negative regular block type etc. These negative versions of blocks should be such that they produce the same results as the original versions if the signs of the tie values were to be reversed.8 Also, null block inconsistency should be corrected to treat both negative and positive deviations from zero as inconsistencies.

7.6.4 Giving more importance to larger inconsistencies in valued blockmodeling

The way of computing inconsistencies within blocks in valued blockmodeling could be changed in such a way that a larger weight would be given to larger deviations (of cell or row/column values from $m$ or 0). For example, these deviations could be squared.

7.6.5 Different computation of inconsistencies for regular blocks in homogeneity blockmodeling

With homogeneity blockmodeling, $f$-regular block inconsistency is the maximum of row-$f$-regular and column-$f$-regular block inconsistencies. From this it follows that the $f$-regular block inconsistency is always equal to either row- or column-$f$-regular block inconsistency,

---

67 Doreian and Mrvar (1996: 161) also presented the criterion function for valued signed networks. Their parts (when confined to only one block) correspond to positive and negative block inconsistencies for valued blockmodeling.

68 The block type inconsistencies for binary blockmodeling are not presented in such a form that it would make sense to compute them on a signed network. However, we can point to the fact that block type inconsistencies for binary blockmodeling correspond to the block type inconsistencies for valued blockmodeling (when used on a binary network) with $m$ equal to 1 and the maximum as function $f$. 
or both if they are the same. This also means that if both conditions (row and column) are broken, only the one where the block type inconsistency is larger affects the \( f \)-regular block inconsistency.

Other options were considered, such as the \( f \)-regular block inconsistency being equal to the sum of row-\( f \)-regular and column-\( f \)-regular block inconsistencies. However, this means that \( f \)-regular block type is not compatible\(^{69} \) with the complete type, even when function \( f \) is the mean.

Another possibility is to ensure that the value from which sum of square or absolute deviations are computed is the same for both row and column \( f \) values when computing \( f \)-regular block inconsistencies. This would mean that in the case of non-zero row-\( f \)-regular and column-\( f \)-regular block inconsistencies the regular block inconsistency would usually be larger than any of them, unless function \( f \) is the mean in sum of squares blockmodeling. However, this would also bring two undesirable consequences. First, it would usually make regular block inconsistency larger than column-\( f \)-regular block inconsistency even when row-\( f \)-regular block inconsistency is 0. Second, it would also make the \( f \)-regular block inconsistency of a block with 0 row-\( f \)-regular inconsistency and identical values in each column higher than the complete block inconsistency of the same block.

7.6.6 Possible problems of implicit blockmodeling solutions using null blocks

Implicit blockmodeling was shown to produce inappropriate blockmodels and consequently often also partitions when used with null blocks. As a possible solution the use of implicit blockmodeling without null blocks was suggested, yet this results in a number of undesired consequences. Another solution would be to move the implicit blockmodeling closer to valued blockmodeling or absolute deviations blockmodeling. The solution described below presumes the use of implicit blockmodeling without a maximum normalization (or \( m \) normalizations, as it should be called based on the modifications suggested below). If this normalization is to be used additional modifications would be necessary.

Both versions include a different way of computing the value from which the deviations (inconsistencies) are computed in all but the null block type. Due to the similarity with valued blockmodeling, let us call it \( m \). Instead of using the maximum (of a block of a row/column), we could use some other function, e.g. a ‘trimmed’ maximum by not taking into account a certain percent of the highest values. In addition, we could specify the lowest allowed value for this value \( m \). If the computed value is lower than this value the block would be declared as null.

The question now remains of what should be done with values over \( m \) (in blocks other than null). One option would be to say that these do not cause any inconsistencies, while the other would be to also treat these deviations from this value as inconsistencies. The first option

---

\(^{69} \)Cannot be used together in the same blockmodel.
makes implicit blockmodeling more similar to value blockmodeling, while the second one makes it more similar to absolute deviations blockmodeling.

Personally I think that the first option is more appropriate, although the second retains more favorable properties of implicit blockmodeling such as the sensitivity of the criterion function to the ‘standard’ (as opposed to definitions specially modified to a certain approach) definitions of structural and regular equivalence. However, it should be kept in mind that these ideas have yet to be tested.

7.6.7 Using valued blockmodeling on binary networks

Valued blockmodeling can also be used on binary networks, even with parameter \( m \) set higher than 1 (in which case this would be binary blockmodeling). This would only be appropriate for \textit{sum}-regular (and similar) blocks. The idea is that we can demand that a block is regular only if there is at least \( m \) 1s (ties) in each row or column (or both).

7.6.8 Using generalized blockmodeling on row and column normalized networks

A similar idea to that suggested in Section 4.4.3 for REGE (based on Nordlund's (2007, 61) ideas for identifying regular blocks in valued networks) can also be easily implemented with generalized blockmodeling. The idea is to simultaneously analyze two networks; one obtained using row and one using column normalization\(^{70}\). The idea is most suited to regular networks where we can demand that an \( f \)-regular block type is ‘comprised’ of a row-\( f \)-regular type in the row-normalized network and a column-\( f \)-regular block type in a column-normalized network. Although the idea seems very reasonable, initial tests indicate that it usually finds partitions where most units are contained in one cluster.

The approach can also be used with other block types. Often we might demand that the same blockmodel (positions of block types) must be present in both networks, or some form of pre-specified blockmodeling can be used.

7.7 Conclusion

The blockmodeling types for the generalized blockmodeling of valued networks presented in this chapter are compared in this section with indirect approaches, binary blockmodeling and amongst themselves. This comparison is only based on their theoretical properties. A more extensive comparison and evaluation is presented Chapter 10 where results based on the application of these approaches to empirical and artificial networks in Chapter 9 and results of the simulations presented in Chapter 10 are also incorporated.

Little new can be said about the comparison of generalized blockmodeling (also for valued networks) and indirect approaches. The benefits of the optimization approach described by

\(^{70}\) Normalized by dividing the values in the row/column by the row/column sum (if greater than 0).
Doreian et al. (2005: 245) are just as relevant for valued networks as they are for binary ones. The most important advantages of generalized blockmodeling over indirect approaches are:

- the value of a criterion function as a measure of fit or the quality of a partition;
- not being limited to just structural and regular equivalence, the possibility to specify (define) equivalence by selecting allowed block types;
- allowing the use of pre-specified blockmodels; and
- the possibility of confirmatory blockmodeling (model fitting).

However, these advantages come at the price. Generalized blockmodeling approaches are much slower. This is especially true of their current implementation in the R package blockmodeling (introduced in Chapter 8).

As the indirect approaches are only developed for structural and regular equivalences, the more direct comparison can be only made for these two special cases.

In the case of valued networks that perfectly comply with structural equivalence, indirect approaches to structural equivalences (provided that the diagonal values in the matrix are handled correctly) provide the same partitions as homogeneity and implicit blockmodeling (according to structural equivalence). In a special case where all non-null values have the same values, the same partitions are also obtained using binary and valued blockmodeling (according to structural equivalence and with an appropriate selection of parameters $m$ and $t$).

In the case of networks that perfectly comply with $max$-regular equivalence, the same partitions are obtained using the ‘One Way’ version of the REGGE algorithm (REGGE-OW) and homogeneity and implicit blockmodeling according to $max$-regular equivalence. Where all row and column maximums within blocks have identical values, binary and valued blockmodeling produce the same result (with an appropriate selection of parameters $m$ and $t$).

In the case where all non-null values have identical values the ‘One Way’ version of REGDI (REGDI-OW, the distance, one-way REGE algorithm) also produces the same result.

Valued blockmodeling was mostly developed on the basis of binary blockmodeling and it is therefore not surprising that it is most similar to binary blockmodeling of all the blockmodeling types for valued networks. Binary and valued blockmodeling were already compared to some extent in Section 7.1, where valued blockmodeling was introduced. There it was also shown that, when applied to a binary network, valued blockmodeling with parameter $m$ set to 1 actually becomes binary blockmodeling. The two remaining blockmodeling types, homogeneity and implicit blockmodeling, are very different from binary blockmodeling. Homogeneity blockmodeling is also very different from valued blockmodeling. Block type inconsistencies are computed very similarly for valued and implicit blockmodeling as was shown in Section 7.3 where implicit blockmodeling was introduced and especially in Subsection 7.3.3. The main difference is that parameter $m$ in valued blockmodeling is replaced by a block maximum (in the case of a row- or column-dominant block by the maximum of the dominant row or column) of the block analyzed. Implicit blockmodeling is, on the other hand, also very similar to homogeneity blockmodeling as they have the same ideal blocks (if when using homogeneity blockmodeling the maximum...
is used as function $f$ in $f$-regular and similar blocks and if the appropriate version of row- and column-dominant blocks is selected – version 2).

Binary and valued blockmodeling were only designed to distinguish different types of blocks or connections (based on patterns of ties). In binary blockmodeling it is only important if the tie is present or not in the binarized version of the network, that is, if the tie value is larger than or equal to slicing threshold $t$ or lower than this value. In valued blockmodeling the situation is similar, only now the difference between the tie value and either 0 or parameter $m$ also matters. As we can see, each of these approaches has a parameter that determines which values are treated as relevant and which as irrelevant. This is usually a disadvantage of these approaches although it can be an advantage if sufficient prior knowledge is available and theory supports such a distinction between relevant and irrelevant ties.

Binary and valued blockmodeling were not designed to distinguish blocks of the same type but with different values. E.g., they were not designed to distinguish between a complete block where all cell values have a value of $a$ and a complete block where all cell values have a value of $b$, where $a \neq b$. This means that binary and valued blockmodeling are unable to find the three clusters presented in Figure 7.2. These two blockmodeling types can either only find the partition 1, 1, 1, 2, 2, 2, 2, 2, 2 or the partition 1, 1, 1, 1, 1, 2, 2, 2, 2. The first partition is optimal if the value of parameter $m$ is larger than 12 when using valued blockmodeling or the value of threshold $t$ used for binarization is greater than 6. Otherwise, the second partition is optimal. In the case of valued blockmodeling, both partitions are optimal if $m$ is exactly 12.

*Figure 7.2: A network with an optimal partition*

On the other hand, homogeneity and impact blockmodeling (and the indirect approaches to structural and regular equivalence) can find the *correct* partition into three clusters. These approaches can also distinguish blocks of the same type with different tie values.
While homogeneity blockmodeling is very suitable for distinguishing blocks of the same type based on tie values, it is less suitable for distinguishing blocks with similar tie values of different types. This is due to the following disadvantages of homogeneity blockmodeling:

1. The null ideal block is defined in the original\textsuperscript{71} version in such a way that it is a special case of any other block type. The null block inconsistency is defined so that it is always greater or equal to the complete block inconsistency. For sum of squares blockmodeling, null block inconsistency is only equal to complete block inconsistency in the case of an ideal null block. The problem is less severe for absolute deviations blockmodeling as there all blocks where at least half cell values have a value of 0 have null block inconsistency equal to complete block inconsistency. Since the null block type is usually given priority, such an empirical block is classified as a null block type. Since complete block inconsistency is usually larger than most other block inconsistencies, there is even less chance that an empirical block is classified as null when these other block types are also allowed. This means that homogeneity blockmodeling is not well suited for the classification of blocks into the null block type as opposed to some other block type.

2. It is questionable if $f$-regular, row-$f$-regular and column-$f$-regular blocks are compatible with other block types unless sum of squares blockmodeling with the mean as function $f$ is used. This especially problematic since the maximum is usually the most appropriate function for function $f$.

Implicit blockmodeling can also distinguish blocks of the same type based on tie values (assuming that block maximums also differ) and it does not have problems with the compatibility of blocks or with the null block being a special case of other block types. However, it has its own set of problems that are mainly caused by the fact that block type inconsistencies are computed relative based on the block maximum. This also means that the block type should be interpreted relative to the block maximum. E.g., an empirical null block with a higher block maximum can have all of its values larger (or equal or larger) than an empirical complete block with a lower maximum. This means that the addition of a large value to a cell in an empirical block previously classified as complete can mean that this empirical block is classified as a null after this addition. An example is presented in Subsection 7.3.3. As block type inconsistencies are very sensitive to changes in block maximums, this also means they are very sensitive to outliers (extremely large values). Of course, anything that affects the classification of blocks also consequently affects the optimal partition returned by the approach. The problem is even more severe if maximum normalization is used since, in that case, the null block inconsistency of an empirical block that is not an ideal null block is actually lowered if the largest value in this block is increased. A possible solution to this problem would be to not use null blocks. However, in this case, implicit blockmodeling practically becomes a version of homogeneity blockmodeling (with a

\textsuperscript{71} Some suggestions were made to tackle this problem.
very unusual measure or variability) and suffers from the same problems, most noticeably from the null block problem.

In the following chapter, the implementation of these approaches in the blockmodeling package (Žiberna, 2007a) for statistical program and environment R (R Development Core Team 2006) is presented. This is followed by an application of these approaches to artificial and empirical examples in Chapter 9 and an evaluation of the approaches on valued regular networks through simulations in Chapter 10. A similar comparison and discussion of the proposed approaches as presented in this section is repeated in Chapter 11 enriched by the results obtained in Chapters 9 and 10.
8 CURRENT IMPLEMENTATION IN R

For testing purposes, evaluation and for use on small networks, most of the approaches suggested in this dissertation have been implemented in the blockmodeling package (Žiberna, 2007a) for statistical program and environment R (R Development Core Team 2006). The implementation is inefficient, meaning that the algorithms are very slow. For comparison, the same algorithms for binary generalized blockmodeling that are implemented in Pajek 1.11 (Batagelj and Mrvar, 2006) run about 1000 times faster.

In addition, some other functions have been created. These are mainly plotting and other convenience functions and a function for computing distances in terms of structural equivalence. A few functions have been written by other authors, which are clearly stated in the help files of those functions. More details about the package and the functions it contains can be found in the help files. Therefore this chapter includes only basic information.

8.1 Generalized blockmodeling

Generalized blockmodeling is implemented for block types in Table 8.1. The function \( f \) used to compute statistics on rows and columns for \((f-)\)-regular blocks (and row- and column-\( f \)-regular) is not presented in the table since several functions can be supplied as an argument (see the previous chapter for the discussion of which functions are appropriate). Networks (matrices) with one or more modes and one or more relational networks are supported. The allowed block types can be specified for the whole matrix or for each block separately.

For homogeneity blockmodeling, the inconsistencies are computed based on the formulas from Table 7.4. The pre-specifying of central values is also supported, while other options from Section 7.4 are not.

The package includes functions for computing the value of a criterion function (and image matrix etc.) for one partition and for several partitions, for optimizing one partition, for optimizing several (possibly randomly chosen) partitions based on the input parameters. Functions for optimizing (one or more) partitions are very slow since they check the criterion function for moving each unit from one cluster to another and for the exchange of each pair of units from different clusters. The algorithm is essentially the same as that suggested by Doreian et al. (2006: 150, 188):

\[
\text{repeat:} \\
\text{if in the neighborhood of the current clustering (partition) } C \\
\text{there exists a clustering (partition) } C' \text{ such that } P(C') < P(C) \\
\text{then move to clustering (partition) } C'
\]

Usually, the neighborhood is determined by two transformations: moving a unit from one cluster to another, and interchanging two units from different clusters.
What is not completely clear is what happens where there are several partitions in the neighborhood of $C$ that have lower values of the criterion functions. Two options are implemented in the blockmodeling package:

1. the one with the lowest value of the criterion function is chosen. If several have the same value, one of them is chosen at random.
2. the first partition $C'$ such that $P(C') < P(C)$ found is chosen.

Since the number of possible partitions is usually enormous, the previously described local search must usually be used. However, for very small networks and only a few clusters sometimes a full search is possible. This means that for all possible partitions (with the desired number of clusters) the value of the criterion function is computed and then the partition(s) with the smallest error is (are) chosen.

Table 8.1: Currently implemented block types by blockmodeling types

<table>
<thead>
<tr>
<th>Blockmodeling type</th>
<th>Binary and valued</th>
<th>Implicit</th>
<th>Homogeneity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implemented block types</td>
<td>Null (&quot;null&quot;)</td>
<td>Null (&quot;null&quot;)</td>
<td>Null (&quot;null&quot;)</td>
</tr>
<tr>
<td></td>
<td>Complete (&quot;com&quot;)</td>
<td>Complete (&quot;com&quot;)</td>
<td>Complete (&quot;com&quot;)</td>
</tr>
<tr>
<td></td>
<td>(f-)regular (&quot;reg&quot;)</td>
<td>Regular (&quot;reg&quot;)</td>
<td>f-regular (&quot;reg&quot;)</td>
</tr>
<tr>
<td></td>
<td>Row-(f-)regular (&quot;rre&quot;)</td>
<td>Row-regular (&quot;rre&quot;)</td>
<td>Row-f-regular (&quot;rre&quot;)</td>
</tr>
<tr>
<td></td>
<td>Column-(f-)regular (&quot;cre&quot;)</td>
<td>Column-regular (&quot;cre&quot;)</td>
<td>Column-f-regular (&quot;cre&quot;)</td>
</tr>
<tr>
<td></td>
<td>Row-dominant (&quot;rdo&quot;)</td>
<td>Row-dominant (&quot;rdo&quot;)</td>
<td>Row-dominant – 3 test versions (&quot;rdo1&quot;, &quot;rdo2&quot;, &quot;rdo3&quot;)</td>
</tr>
<tr>
<td></td>
<td>Column-dominant (&quot;cdo&quot;)</td>
<td>Column-dominant (&quot;cdo&quot;)</td>
<td>Column-dominant – 3 test versions (&quot;cdo1&quot;, &quot;cdo2&quot;, &quot;cdo3&quot;)</td>
</tr>
<tr>
<td></td>
<td>Row-functional (&quot;rfn&quot;)</td>
<td>Row-functional (&quot;rfn&quot;)</td>
<td>Row-functional (&quot;rfn&quot;)</td>
</tr>
<tr>
<td></td>
<td>Column-functional (&quot;cfn&quot;)</td>
<td>Column–functional (&quot;cfn&quot;)</td>
<td>Column–functional (&quot;cfn&quot;)</td>
</tr>
<tr>
<td></td>
<td>Density (&quot;d&quot;), Average (&quot;avg&quot;)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the examples given in the next chapter both approaches were used. A full search was used for some of the smallest examples presented and a local search with only a few starting partitions for others. Where a local search was used (all possible partitions were not checked), there is no guarantee that the partition with the smallest inconsistency found is truly the
globally optimal partition (the partition with the smallest inconsistency among all possible partitions). Therefore, quotation marks were used in the paper to indicate that those partitions named ‘optimal’ may be only locally optimal. When a full search was used, the quotation marks were not used.

8.2 Indirect approaches

In addition to generalized blockmodeling, some indirect approaches to blockmodeling were also implemented. They were initially just meant for a comparison and for a quick analysis prior to the analysis with generalized blockmodeling. However, some new versions of REGE were then also included.

As mentioned above, a function that computes differences in terms of structural equivalence has also been included for convenience. Several possibilities for treating the diagonal values of the matrix in one-mode networks are implemented. For two-mode networks, the indirect approach can be used on one mode at a time using functions for computing the distances included in other packages in the R program.

Most versions of REGE presented in the section on indirect approaches for regular equivalence (Section 4.4) are implemented. These include those found in the previous literature and those developed in this dissertation. The exceptions are some versions of REGE modified for networks where the tie values are influenced by the ‘size’ of the units (e.g. trade networks among countries and networks of carbon flow among species), as presented in Subsection 4.4.3, such modifications are only implanted for the REGGE algorithm and its ‘One Way’ version.

8.3 Other functions

Several other functions have been included in the package. They can be grouped in the following categories:

- plotting functions;
- functions for creating and counting all possible partitions of n units into k clusters (written by Chris Andrews);
- functions for reading and writing Pajek files (some written by Vladimir Batagelj);
- functions for the extraction of parts of objects, returned by functions for generalized blockmodeling; and
- other utility functions.
This chapter has two purposes. The first is to demonstrate the application of the presented approaches to real networks. The second is to compare the proposed approaches through their application to mainly empirical but also some artificial networks.

Each example is chosen for a specific purpose. While these purposes are described in more detail at the start of the corresponding sections, they are also outlined here. With the ‘Simple artificial network’ I wish to demonstrate the basic working of the generalized blockmodeling approach and the computation of the criterion function. The next example, the ‘Artificial REGE examples’, are designed to show the differences among several versions of REGE. The rest of the examples are empirical networks. The first of these is ‘The cooperation of occupational social workers with other organizations’. This example was chosen as an example of an undirected network. It is a one-mode network, obtained from a two-mode network, although the construction of the networks is not of interest here. Also, as the network is very small a full search is feasible. The next example is the network of ‘Borrowing notes between social-informatics students’. This is an example of a directed and still relatively small network. This is also the only example of a social network presented in this chapter as only here do the units represent people. It is also the most thoroughly examined example in this chapter. This example is followed by the network of an island ‘sweet’ from EAT (The Edinburgh Associative Thesaurus’). This is an example of a larger network where a full search is unfeasible. Further, the network is also measured on a ratio (and therefore also an interval) level of measurement. Due to the distribution of the tie values the network is also suitable for the most generalized blockmodeling approaches presented in this dissertation, the only exception being implicit blockmodeling (with the null block type). The last example is a food web, more precisely the network based on ‘St. Marks carbon flow data’. It provides a nice example of a network where tie values are proportional to the ‘size’ of the units. For such networks, modified versions of REGE presented in Section 4.4.3 were designed and their application is presented through this example.

In the examples below, the results of several methods are often compared. Unfortunately, there are usually no quantitative criteria which can be used for such comparisons. Of course, each (generalized) blockmodeling type (with a selection of allowed block types or a blockmodel) defines a criterion function that could be used as a criterion for comparing the partitions obtained using different approaches. However, the approach that was used to define the criterion function will surely produce the partition with the lowest value of the criterion.
function. Although other approaches may match this (lowest) value, none can attain a lower value\(^{72}\). Therefore, if we select such a criterion we have also already determined which approach will produce the best partition. Even if some other criterion would have been constructed that would best measure the appropriateness of the partition, such a criterion would be best used in the optimization. Therefore, quantitative criteria are not suitable for a comparison of the partitions obtained through the optimization of different criteria functions.

Due to the lack of quantitative criteria, the obtained partitions were mainly compared based on the appearance of the partitioned matrices. This inherently subjective judgment was mainly made on the basis of how well the empirical blocks corresponded to a description of the ideal blocks of block types used in the optimization process. Especially undesired were empirical blocks that were somewhere in between the ideal null block and some other ideal block (e.g. ideal complete or ideal \((f)\)-regular block). Also, partitions where one cluster contained most units were evaluated as being less satisfactory.

In addition, the results of different approaches were sometimes compared based on the interpretability of the obtained blockmodels. This aspect was, however, considered less important due to the following two reasons. First, I am not an expert in any field from which the examples were taken. Although some (e.g. the ‘borrowing notes’ and ‘cooperation among social workers’ examples) are more familiar to me than others (e.g. the ‘St. Marks food web’ example), I am not qualified to judge which blockmodel provides a more reasonable interpretation. Second, the interpretability of the obtained solutions does not offer good grounds for a comparison of the approaches.

### 9.1 Example 1: Simple artificial example

This first example was constructed for several reasons. First, it is used to demonstrate the computation of the criterion function on a very simple example. Second, it shows some properties of valued blockmodeling and binary blockmodeling. These are that if a partition has the value of the criterion function for valued blockmodeling at \(m = x\) equal to 0, then the value of the criterion function for valued blockmodeling is also equal to 0 for all values of parameter \(m\) lower than \(x\). The same partition also has the value of the criterion function for binary blockmodeling (where all ties are treated as ‘1’ ties). In cases where the maximum (this is not the case here) is used as function \(f\) in \(f\)-regular and similar block types, the same partition has the value of the criterion function for binary blockmodeling with the slicing threshold \(t \leq x\).

Also, we wish to show that several methods and options within these methods can often produce the same results.

\(^{72}\) Of course, when using a local search there is no guarantee that the approach whose criterion function is used as the criterion will find the globally optimal partition. Let us call this approach the defining approach. In such cases, other approaches can find partitions with lower values of the criterion function. However, if these partitions are used as starting points for local optimization using the defining approach, the defining approach would also find these partitions or partitions with even lower values of the criterion function.
The simple artificial network is presented in Figure 9.1. It was specially designed to fit the blockmodel for valued blockmodeling with parameter $m = 3$, partition $1, 1, 1, 2, 2, 2$ (later in this section referred to as the original partition), function $f = \text{‘sum’}$ and a blockmodel:

```
1 2
1 "com" "reg73"
2 "reg" "null"
```

This model was also used as a pre-specified blockmodel when pre-specified blockmodeling was used.

*Figure 9.1: Simple artificial network*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In this special case, due to the small number of units and only two clusters the criterion function can be computed for all possible partitions (a full search) in a very short time.

### 9.1.1 Binary blockmodeling

Binary blockmodeling produces the original partition (in Figure 9.1) with the slicing threshold $t = 1$ and the pre-specified model (presented above). Here the value of the criterion function is 0. For the slicing threshold $t = 2$, the optimal partition is the partition $1, 1, 2, 2, 2, 2$. Let us use this example (binary blockmodeling with a slicing threshold $t = 2$) to demonstrate the computation of the criterion function for binary blockmodeling for the original partition. For higher values of slicing threshold $t$, one cluster contains just one unit.

The block $R(1, 1)$ is pre-specified as complete (in the pre-specified blockmodel). We see in Figure 9.1 that it is indeed an ideal complete block since it has all values (except the diagonal ones, that can be different) larger than 2. The (complete) block inconsistency for block $R(1,1)$

---

73 The "reg" in the case of valued blockmodeling stands for ‘$f$-regular’ block. The exact specification of the block depends on the function ‘$f$’.
is therefore 0. Similarly, block $R(2, 2)$ is pre-specified as null and it has all cell values equal to 0. As such, it has (null) block inconsistency equal to 0.

Block $R(2, 1)$ is also an ideal null block (as it has no tie values equal to or greater than 2), yet it is pre-specified as regular. Regular block inconsistency in binary blockmodeling is computed as $(n_c - p_c)n_r + (n_r - p_r)p_c$. $n_c$ and $n_r$ are the number of columns and the number of rows and are both 3. $p_r$ and $p_c$ are numbers of non-null rows and columns (numbers of rows and columns with at least one tie value equal to or greater than 2) in a block and are both 0. Therefore, the block $R(2, 1)$ (regular) block inconsistency is $(3 - 0)*3 + (3 - 0)*3 = 9$. The last remaining block is block $R(1, 2)$, which is also pre-specified as regular. Therefore, the same formula is used as before. $n_r$ and $n_c$ are the same as before, 3, while $p_c$ is 3 and $p_r$ is 2. The (regular) block inconsistency is therefore $(3 - 3)*3 + (3 - 2)*3 = 3$. The total inconsistency of the network is the sum of these block type inconsistencies (by rows): $0 + 3 + 9 + 0 = 12$.

Binary blockmodeling according to structural equivalence results in an original partition with the slicing threshold = 1, 2, 3. The models (images) obtained are:

<table>
<thead>
<tr>
<th>Slicing threshold $t = 1$</th>
<th>Slicing threshold $t = 2$ or 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>&quot;com&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;com&quot;</td>
</tr>
</tbody>
</table>

Binary blockmodeling according to regular equivalence with the slicing threshold $t = 1$ finds the original partition as one of 9 optimal partitions with a value of the criterion function equal to 0. In the case of a slicing threshold $t = 3$, binary blockmodeling also finds the original partition as the optimal with the value of the criterion function 1. The models returned are the same for the original partition as with the structural equivalence, except the complete blocks are replaced by regular blocks. If the slicing threshold $t = 2$ is used, the partition with units 1 and 2 in the first cluster is returned. If the slicing threshold $t$ is higher than 3 when using binary blockmodeling according to either structural or regular equivalence only two ties remain and all partitions have equal inconsistency.

### 9.1.2 Valued blockmodeling

Since the network was designed for the function sum over rows and columns, only this function was used for valued blockmodeling and only the mean for homogeneity blockmodeling. Valued blockmodeling according to pre-specified blockmodeling, $f = \text{‘sum’}$ and $m = 1, 2, 3$ or 4 results in the optimal partition. When the value of parameter $m$ is 1, 2 or 3, the total inconsistency (of the original partition - 1, 1, 1, 2, 2, 2) is 0. For $m = 4$, the value of the criterion function is 21. For $m = \text{lower than or equal to 7}$, units 1 and 2 are in the first cluster. For $m$ larger than or equal to 8, only unit 1 remains in the first cluster.

$^{74}$ The use of the function mean over the rows and columns in homogeneity blockmodeling is analogous to the use of the function sum in valued blockmodeling.
Valued blockmodeling according to the pre-specified model and $m = 4$ is used to demonstrate the computation of the criterion function for valued blockmodeling. The block $R(1, 1)$ is pre-specified as complete. The complete block inconsistency (for a diagonal block) is computed as:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} (m-b_{ij})^+ + \min\left(-\sum(m-diag(B))^+ + \sum diag(B), 0\right)$$

The off-diagonal part of the inconsistency is computed as $\sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} (m-b_{ij})^+$, while the diagonal part is equal to $\min\left(\sum(m-diag(B))^+, \sum diag(B)\right)$. Therefore, as all values on the diagonal are 0 the inconsistencies on the diagonal are also all 0.

The off-diagonal inconsistencies are computed as $m - b_{ij}$. The cell inconsistencies are therefore:

<table>
<thead>
<tr>
<th></th>
<th>[,1]</th>
<th>[,2]</th>
<th>[,3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>[2,]</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>[3,]</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The (complete) block inconsistency for block $R(1, 1)$ is therefore 4. Block $R(2, 2)$ is pre-specified as null. As all values are 0 and the null block inconsistency is computed as the sum of all the tie values in the blocks, the (null) block inconsistency for block $R(2, 2)$ is 0.

Blocks $R(2, 1)$ and $R(1, 2)$ are pre-specified as sum-regular. The sum-regular block inconsistency is computed as:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \max\left((m - \Sigma(B_{[i]}))^+, (m - \Sigma(B_{[j]}))^+\right)$$

The inconsistency of an individual cell in the sum-regular block is computed as the difference between $m$ and either the row sum or column sum of the row or column to which the cell belongs. Whichever difference is larger is used. If the larger one is also negative the cell inconsistency is 0. The cell inconsistencies computed in such a form are:

**Block $R(2, 1)$**

<table>
<thead>
<tr>
<th></th>
<th>[,1]</th>
<th>[,2]</th>
<th>[,3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>[2,]</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>[3,]</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Block $R(1, 2)$**

<table>
<thead>
<tr>
<th></th>
<th>[,1]</th>
<th>[,2]</th>
<th>[,3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>[2,]</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>[3,]</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Therefore, the (sum-regular) block inconsistency is 9 for block $R(2, 1)$ and 8 for block $R(1, 2)$. This makes the total inconsistency of the network with the partition 1, 1, 1, 2, 2, 2 computed using valued blockmodeling with $m = 4$ according to the pre-specified blockmodel (by rows) $4 + 8 + 9 + 0 = 21$.

For structural equivalence and $m = 1, 3, 4, 5$ or 6 also resulted in the optimal partition. The total inconsistencies were 3, 19, 23, 28 and 34, respectively. The blockmodels obtained are:
Parameter $m = 1$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;com&quot;</td>
<td>&quot;com&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;com&quot;</td>
<td>&quot;null&quot;</td>
</tr>
</tbody>
</table>

Parameter $m = 2, 3, 4, 5, 6$ or $7$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;com&quot;</td>
<td>&quot;null&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;null&quot;</td>
<td>&quot;null&quot;</td>
</tr>
</tbody>
</table>

For $m = 2$, only units 2 and 3, and for $m = 7$, only units 1 and 3 compose the first cluster. For $m$ higher or equal to 8, all the partitions have the same inconsistencies.

With sum-regular equivalence (+ complete blocks and still valued blockmodeling), values of $m = 1, 2, 3, 5, 6, 7$ and $8$ produce the original partition with values of the total inconsistency $0, 0, 19, 19, 24$ and $32$, respectively. When $m = 1$, $8$ other (than the original) partitions are also found. Three different models were obtained depending on the value of $m$.

Parameter $m = 1$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;com&quot;</td>
<td>&quot;reg&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;com&quot;</td>
<td>&quot;null&quot;</td>
</tr>
</tbody>
</table>

Parameter $m = 2$ or $3$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;com&quot;</td>
<td>&quot;reg&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;reg&quot;</td>
<td>&quot;null&quot;</td>
</tr>
</tbody>
</table>

Parameter $m = 5, 6, 7$ or $8$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;reg&quot;</td>
<td>&quot;null&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;null&quot;</td>
<td>&quot;null&quot;</td>
</tr>
</tbody>
</table>

For the higher values of $m$ (from $9$ upward), all the partitions produce the same total inconsistency, $40$. This is equal to the sum of all cell values as with such high values of parameter $m$ the obtained blockmodel consists only of null blocks.

We note that the total inconsistency never decreases as parameter $m$ increases. It can only increase or stay the same. This is a general property of valued blockmodeling.

**9.1.3 Implicit blockmodeling**

For implicit and homogeneity (sum of squares and absolute deviations) blockmodeling the original partition is the optimal partition for all versions tested. The versions refer to the use of pre-specified blockmodeling, structural equivalence and regular equivalence. For implicit blockmodeling, structural and regular equivalence were tested with and without null blocks and with and without normalizations (maximum and block size). The computation of the criterion function is presented on the case of pre-specified blockmodeling (as before) for each (major$^{75}$) blockmodeling type separately.

The first computation of the criterion function for implicit blockmodeling is demonstrated on the non-normalized version. Block $R(1, 1)$ is classified as complete. The complete block inconsistency for implicit blockmodeling (without normalization) is computed as:

$$\sum (\max \{B \neq 0\} - B) + \min(0, \sum \text{diag}(B) - \sum (\max \{B \neq 0\} - \text{diag}(B)))$$

$^{75}$ Only one representative of homogeneity blockmodeling is chosen.
In our case, where the diagonal contains only zeros this reduces to 
\[
\sum_{i=1}^{n_i} \sum_{j=1}^{n_j} (\max\{B \neq 0\} - b_{ij}),
\]
that is the sum of deviations of the cell values from the block maximum for non-diagonal cells. The block maximum is 5 in block \(R(1, 1)\). The deviations are therefore:

\[
\begin{array}{ccc}
[1,1] & [1,2] & [1,3] \\
[1,1] & 0 & 2 & 0 \\
[2,1] & 1 & 0 & 2 \\
[3,1] & 2 & 2 & 0 \\
\end{array}
\]

The (complete) block inconsistency of block \(R(1, 1)\) is therefore 9.

The blocks \(R(2, 1)\) and \(R(1, 2)\) are pre-specified as regular. The regular block inconsistency in implicit blockmodeling is computed as:

\[
\sum_{i=1}^{n_i} \sum_{j=1}^{n_j} \max\{\max\{B \neq 0\} - \max\{B_{[i,k]}\}, \max\{B \neq 0\} - \max\{B_{[i,j]}\}\}
\]

The regular block inconsistency is computed as the sum of cell inconsistencies, which are computed as the maximums of the deviations of the row and column (to which a cell belongs) maximums from the block maximums. In block \(R(2, 1)\) is actually an ideal complete block with a block maximum of 1. As such, it is also an ideal regular block since all row and column maximums are equal to the block maximum. The block \(R(1, 2)\) is more interesting. The block maximum here is 3. Therefore, the cell inconsistencies for this block are:

\[
\begin{array}{ccc}
[1,1] & [1,2] & [1,3] \\
[1,1] & 1 & 1 & 0 \\
[2,1] & 1 & 1 & 1 \\
[3,1] & 2 & 2 & 2 \\
\end{array}
\]

The (regular) block inconsistency of block \(R(1, 2)\) is therefore 11.

The last block \(R(2, 2)\) is pre-specified as null and it is an ideal null block. Its (null) block inconsistency is therefore 0. The null block inconsistency is computed as the sum of all cell values (deviations from 0). If all values on the diagonal are not 0 the inconsistencies of these values can be computed from the block maximum (only in diagonal blocks, which block \(R(2, 2)\) is). Whichever computation produces lower block inconsistency is used (however inconsistencies for all diagonal cells in a given block must be computed in the same way).

The total inconsistency for implicit blockmodeling according to pre-specified blockmodeling and without any normalization is (by rows) \(9 + 11 + 0 + 0 = 20\).

9.1.4 Homogeneity blockmodeling

As the computation of inconsistencies is very similar for sum of squares and absolute deviations blockmodeling, the computation is only demonstrated on absolute deviations blockmodeling. In the case of sum of squares blockmodeling, the difference is that the mean is used as a measure of the central value (instead of the median) and squared deviations from
that value are used instead of absolute deviations. Again, the computation is demonstrated on
the case of pre-specified blockmodeling and the function mean is used as function $f$.

The block $R(1, 1)$ is pre-specified as complete. The complete block inconsistency is computed
as: $\sum_{i\neq j} (b_{ij}) + \text{diag}(B)$

That is, the complete block inconsistency is the sum of the absolute deviations of the cell
values from their median (for diagonal blocks such as $R(1, 1)$, the absolute deviations are
computed separately for off-diagonal and diagonal cells. In the block $R(1, 1)$, the diagonal
cells are all equal (0), and therefore also equal to their median. The sum of their absolute
deviations is therefore 0.

The off-diagonal cell values are (sorted by size) 3, 3, 3, 3, 4 and 5. Their median is 3.
Therefore, the absolute deviations are 0, 0, 0, 0, 1 and 2, and their sum is 3. This makes 3 the
(complete) block inconsistency of block $R(1, 1)$.

The blocks $R(2, 1)$ and $R(1, 2)$ are pre-specified as mean-regular. The mean-regular block
inconsistency is computed as: $\max\left(\sum_{i=1}^{r} \text{mean}(B_{i,j})n_i, \sum_{j=1}^{c} \text{mean}(B_{i,j})n_j\right)$

That is, the mean-regular block inconsistency equal to the maximum of absolute deviations
from the median of the row and column means, multiplied by the number of elements in a row
or column, respectively.

For block $R(2, 1)$, all row and all column means are 1. Obviously, the sum of absolute
derivations is therefore 0 for both rows and columns which makes the (mean-regular) block
inconsistency of block $R(2, 1)$ 0. The computation of the (mean-regular) block inconsistency
of block $R(1, 2)$ is more interesting. The row means of block $R(1, 2)$ are 1, 1.33 and 1. Their
median is 1, the absolute deviations from it are 0, 0.33 and 0, while their sum is 0.33. This
must be multiplied by the number of elements in each row, which is 3. This makes the
inconsistency based on row means $3 \times 0.33 = 1$ (0.33 is an approximation of $1/3$).

The column means are 1, 1 and 1.33, which are essentially the same values as the row means,
just in a different order. As the number of elements in each column is also 3, this makes the
inconsistency based on the columns equal to the inconsistency based on rows, which is 1. The
maximum of these two 1s is also 1. Therefore, the (mean-regular) block inconsistency of
block $R(2, 1)$ is 1.

The last block $R(2, 2)$ is pre-specified as null. The null block type inconsistency for absolute
deviations blockmodeling is computed as the sum of absolute deviations from 0. All values in
block $R(2, 2)$ are 0 so it is an ideal null block and its (null) block inconsistency is 0. The total
inconsistency for absolute deviations is therefore (by rows) $3 + 1 + 0 + 0 = 4$.

### 9.1.5 Conclusion

In this section, the computation of the criterion function (or inconsistencies) was
demonstrated for all blockmodeling types. As this was an artificial and for some methods an
ideal example, all of the blockmodeling types found the original partition with the correct parameters. The advantage of implicit and homogeneity blockmodeling is shown in that they do not need any additional parameters. One of the problematic aspects of using binary blockmodeling was also shown. In the case of using relatively small values of the slicing threshold $t$ (only $t = 1$ in this example) and not very stringent equivalences (like regular equivalence), there is a danger of obtaining a large number of partitions with the same inconsistency.

### 9.2 Example 2: Artificial REGE examples

In an attempt to clarify the differences among the different REGE algorithms, several artificial networks are constructed and analyzed using REGE. This is not necessary for direct approaches as they have clearly defined ideal blocks. In addition, binary and valued blockmodeling are not suited to such examples, where we cannot select one slicing parameter or one value of the parameter $m$ that is suitable for all blocks. Implicit and homogeneity blockmodeling (with max-regular equivalence) are very suitable for such examples; however, in these idealized settings they would always produce the same partitions as REGGE-OW. The exception occurs if we seek a regular equivalence partition that does not correspond to the maximal regular equivalence. All versions of REGE are only suitable for finding (an approximation to) maximal regular partitions, while generalized blockmodeling approaches can also find other regular partitions.

The network in Figure 9.2 is a very simple network. It is a classical core-periphery model. The network has a clear partition into core units (1-4) and periphery units (5-10). The network is undirected since each arc has a reciprocal arc. The matrix representing it is therefore symmetric. If this were a binary network (with all non-null values having a value of 1), all versions of REGE described above would find that all units together form one cluster (one regular equivalence class). However, in the valued case, all versions produce two clusters that can be seen in Figure 9.2. In the matrix representation the clusters are separated by a thicker black line, while in the graphic representation they are distinguished by vertex colors (white – core, black – periphery).

Network 1 is symmetric. Network 2 is very similar to Network 1. However, the ties among the core (1-4) units and units 5-7 (3 of the periphery units) are no longer symmetric. The periphery cluster is split into two clusters, as indicated by the additional gray line on the matrix in Figure 9.3. The difference between these two clusters is best seen in the graph in Figure 9.3. The black vertices are unchanged and have both an incoming and an outgoing arc with at least one of the white vertices. The gray vertices are also joined with the white vertices through incoming and outgoing arcs; however, no gray vertex has both an incoming and an outgoing arc with the same white vertex.

---

76 The gray line indicates a split, found by only some versions of REGE.
Here the difference between those algorithms where symmetry matters and those where it does not (all ‘one way’ algorithms, marked with OW in the previous section) comes into focus. The ‘one way’ (or OW) algorithms find the same clusters as in network 1 (only the split indicated by the black line in the matrix in Figure 9.3), while the other approaches find three clusters presented in Figure 9.3 (the additional split is indicated by the gray line).

The second difference among the regular partitions that these algorithms seek corresponds to which values have to be equal in order for the units to be considered regularly equivalent. For REGGE (when the symmetry requirement mentioned above is satisfied) and REGGE-OW algorithms, in each block only the row and column maximums have to be equal. On the other hand, for the other four versions of REGE, in each block each row and each column have to contain the same values (ignoring zeros), where each unique value can appear one or more times. Network 1, modified in a way so that this effect can be shown (Network 3), is
presented in Figure 9.4. In this network, the REGGE and REGGE-OW algorithms find the same two clusters as shown in Figure 9.2 and are unaffected by the addition of the arcs with a value of 2 to blocks $R(2,1)$ and $R(1,2)$.

*Figure 9.4: Network 3*

![Network 3 diagram]

The other algorithms find the three clusters presented in Figure 9.4. The black vertices are the same as they were in Figure 9.2, while the gray vertices have additional two-way arcs with the value 2 linking them to the white vertices.

The same difference in the characteristics of the REGGE, REGGE-OC and REGDI algorithms can also be shown for network 4 in Figure 9.5, although blocks $R(2,1)$, $R(3,1)$, $R(1,2)$ and $R(1,3)$ still contain only arcs with a value of 5. However, in network 4 block $R(2,1)$ contains the arcs from network 1 and network 2 (from the same block). Each gray vertex now has at least one unreciprocal arc from a white vertex and at least one reciprocal tie with a white vertex. The one versus two arcs in a dyad now take the place of different values. The REGGE algorithm only finds two clusters in this network, while the REGDI and REGGE-OC algorithms find the three clusters shown in Figure 9.5.

In this example I have demonstrated the difference among the REGE algorithms. There are two main characteristics on which these algorithms differ in terms of what kind of partitions they seek. The first difference is in whether or not they evaluated each arc separately or jointly with the arc between the same two units in the opposite direction, if such an arc exists. That is, if they treat two potential arcs in a link between two units jointly or separately. In the case of undirected networks or if all arcs are unreciprocal, the algorithms that only differ on these characteristics seek the same ideal partitions.

The other characteristic that distinguishes the REGE algorithms is which tie values have to be equal in order for two units to be declared regularly equivalent. Some algorithms demand that if unit $a$ has ties with values $x$, $y$ and $z$ to the units in cluster $C$, then unit $b$ can only be equivalent to unit $a$ if it also has ties with values $x$, $y$ and $z$ to the units in cluster $C$. However, it does not matter how many ties with any of these values either unit $a$ or unit $b$ has to the
units in cluster C or if they are connected to all or only some units in cluster C. The other algorithms are less strict and demand that if unit $a$ has ties with values $x$, $y$ and $z$ to the units in cluster C, then unit $b$ can only be equivalent to unit $a$ if it also has a tie with a value equal to the maximum of $x$, $y$ and $z$ to a unit in cluster C. Of course, for those algorithms that treat two potential arcs in a link jointly this must also be taken into account, as demonstrated in the above examples.

Figure 9.5: Network 4

9.3 Example 3: The cooperation of occupational social workers with other organizations

The data come from a survey among occupational social workers in Slovenia (Rape, 2005: 75). The vertices represent organizations with which at least two occupational social workers cooperate. The value of the tie between two organizations is the number of social workers who cooperate with both of them. The diagonal values were set to 0. In Figure 9.6 the network and a ‘hand-picked’ partition are presented.

The example was selected for several reasons. First, it is an example of a small, undirected network. Therefore, a full search was feasible, allowing me to be sure that the partitions found are truly optimal (with respect to the selected criterion function). The second reason is of a more practical nature. Namely, I observed the original research from which these data were obtained and am with contact with the researcher who conducted that research. Therefore, with the help of that researcher I am also able to evaluate the results of the blockmodeling.

Another option would be to put the number of social workers who cooperate with a certain organization on the diagonal. This information was not put on the diagonal to limit its influence on the partition.
analysis from a substantive prospective. In addition, a ‘hand-picked’ partition is available that can be used as an example of a good partition.

*Figure 9.6: Matrix of the cooperation of occupational social workers with other organizations and a ‘hand-picked’ partition*

The ‘hand-picked’ partition separates the organizations into three clusters, the ‘core’ organizations with which most of the social workers cooperate, and two unconnected clusters (there are no connections between these two clusters). These clusters contain the remaining health care organizations and the ‘social environment’. Even in this figure, we can see that the ‘Disability commission’ does not really fit anywhere (in the three-cluster solution). It is the only unit that is not connected to the core units, but only to the ‘Medicine of work’. The other ‘problematic’ unit is ‘SWC’. ‘SWC’ is ‘leaning’ towards the core cluster, however it does not belong there since it has no tie to the ‘other healthcare organizations’ cluster.
9.3.1 Indirect approaches

From Figure 9.7 it can be seen that both structural equivalence\(^78\) and REGGE\(^79\) provide similar results as the hand-picked partition when applied to a valued network. If we compare the three-cluster partitions that are evident from these two dendrograms, we see that we only need to either move only one unit to another cluster (REGGE) or exchange two units (structural equivalence) to obtain the ‘hand-picked’ partition. Also, the movement of only one unit is required to obtain the REGGE partition from the structural equivalence partition or vice versa. However, the structural equivalence dendrogram suggests only two clusters, while the REGGE one suggests two to four clusters.

**Figure 9.7: Dendrograms obtained by Ward's hierarchical clustering on (dis)similarities computed using structural equivalence and REGGE (regular equivalence) computed on a valued network**

In an attempt to show the affects of binarization, the indirect approach according to structural equivalence was also applied to a binarized network. The dendrogram obtained is presented in Figure 9.8. The network was binarized in such a way that all ties in the valued network also resulted in ties in the binary network (the threshold \(t = 1\) was used in the binarization). REGGE was not applied to this network since for a binary symmetric network the maximal regular partition (which REGGE finds) is always trivial (a one-cluster partition).

Based on this dendrogram we can see that for a binarized network, three is an appropriate number of clusters also according to structural equivalence. The three-cluster partition

\(^78\) Measured by Corrected Euclidean-like dissimilarity (Burt and Minor, 1983 in Batagelj et al., 1992: 71).

\(^79\) White’s (1985a) REGGE algorithm (see Subsection 4.4.1 for details) with 10 iterations is used. The one-way version of REGGE would produce identical results since the network is not directed. REGDI produced almost identical results as REGGE.
indicated by this dendrogram is very similar to the hand-picked partition. The only difference is that ‘Health’ has moved from the core cluster to what can now be called the ‘healthcare’ cluster.

Figure 9.8: Dendrograms obtained by Ward’s hierarchical clustering on dissimilarities computed according to structural equivalence on a binarized network

9.3.2 Generalized blockmodeling – structural equivalence

As we see from the dendrogram based on dissimilarities computed according to structural equivalence in a valued network, the distinction between the core and the rest of the units is very clear, while the gap between the other two clusters is small.

The homogeneity and implicit blockmodeling\(^{80}\) confirm this since all these approaches did find a good two-cluster solution, while most have problem finding a good three-cluster partition. Most of the approaches found exactly the core cluster indicated by the dendrogram and the ‘hand-picked’ partition, while only the absolute deviations approach also included ‘SWC’ in the core cluster.

Only implicit blockmodeling provided a good three-cluster partition. When using only complete blocks, the obtained partition was the ‘hand-picked’ partition. The partition obtained allowing both null and complete was very similar, with the only difference being that the ‘Disability commission’ that moved to the other non-core cluster. This is also the only approach (within homogeneity and implicit blockmodeling) that produces a blockmodel (the classification of blocks into null and complete), however this blockmodel is unsatisfactory. The partition is presented in the first matrix in Figure 9.9. The blockmodel is indicated by the

\(^{80}\) Four different versions were considered: sum of squares, absolute deviations and implicit blockmodeling with only complete blocks and implicit blockmodeling with null and complete blocks.
use of shading instead of solid color when indicating the tie strength for identification of the null blocks. It seems unreasonable that block $R(1, 2)$ is null, while block $R(2, 2)$ is complete.

**Figure 9.9: Matrices partitioned in terms of structural equivalence**

<table>
<thead>
<tr>
<th>Implicit blockmodeling null and complete</th>
<th>Absolute deviations blockmodeling complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDII</td>
<td>ESS</td>
</tr>
<tr>
<td>PDII</td>
<td>11</td>
</tr>
<tr>
<td>ESS</td>
<td>11</td>
</tr>
<tr>
<td>Health</td>
<td>9</td>
</tr>
<tr>
<td>Municipality</td>
<td>2</td>
</tr>
<tr>
<td>Schools</td>
<td>2</td>
</tr>
<tr>
<td>Dis. comm.</td>
<td>4</td>
</tr>
<tr>
<td>SWC</td>
<td>1</td>
</tr>
<tr>
<td>NGO</td>
<td>3</td>
</tr>
<tr>
<td>Med. of work</td>
<td>3</td>
</tr>
<tr>
<td>Psy., Addict.</td>
<td>3</td>
</tr>
</tbody>
</table>

For approaches that do not include null block type the idea is to avoid strictly classifying blocks into null or complete, but to assign a value to them$^{81}$. For complete blocks, one$^{82}$ suitable value is simply the mean of the block values (ignoring the diagonal here since there are only zeros). In such a way we can obtain the following valued matrix as a representation of the reduced graph for implicit blockmodeling using only complete (‘hand-picked’ partition) blocks and three clusters:

$$
\begin{bmatrix}
[1,] & [2,] & [3,] \\
1 & 8.333333 & 1.666667 \\
2 & 1.666667 & 1.000000 \\
3 & 1.444444 & 0.000000 \\
\end{bmatrix}
$$

This information can also be presented in the form of an image graph, as seen in Figure 9.10. We see that the three core organizations are strongly connected to each other and very weakly to the both peripheral clusters. These periphery clusters also have light connections within them and no connections to the other periphery cluster.

---

$^{81}$ If we are using several block types (not just complete, but e.g. also regular, row-functional etc.), we also classify empirical blocks into block types within these approaches. However, this classification is not used to distinguish null block type and other block types but only to distinguish among other block types (other than null).

$^{82}$ See Section 7.4 for the other options.
If we want to reduce the number of ties or to obtain a blockmodel (with null and complete blocks), we can select a threshold and treat all values lower than the threshold as zeros or null blocks (depending on the aim we have in mind). Another way of obtaining a blockmodel is to use valued blockmodeling with a certain selected value of \( m \) to classify blocks. In this example, we can conclude that all the ties are relevant, even those with a value of 1. Therefore, \( m \) is set to 1. The image we obtain by using \( m = 1 \) on the ‘hand-picked’ partition in Figure 9.6 is:

\[
\begin{bmatrix}
1,1 & 1,2 & 1,3 \\
[1,] & "com" & "com" & "com" \\
[2,] & "com" & "com" & "null" \\
[3,] & "com" & "null" & "com"
\end{bmatrix}
\]

The image corresponds to the image graph in Figure 9.10. However, I believe it is better to characterize the connections between the different clusters by different tie strengths than by different block types only, as additional information is then provided.

Both homogeneity blockmodeling approaches do not provide such a good three-cluster partition, as they split the core cluster and not the periphery cluster into a two-cluster partition. If the number of clusters is increased to four, these two approaches again perform better. The partition obtained using sum of squares blockmodeling is essentially the same as the ‘hand-picked’ partition, the only difference being that ‘PDII’ is in its own cluster. The partition obtained by absolute deviations is slightly different and is presented in the second matrix in Figure 9.9. The partition is similar to the sum of squares partition, only that the units ‘SWC’ and ‘Disability commission’ are exchanged.

At the end of the previous subsection, I showed that using structural equivalence methods on binarized network also leads to an appropriate partition. Therefore, binary blockmodeling was also applied to the network binarized with a threshold \( t = 1 \). Two partitions are obtained as optimal using binary blockmodeling according to structural equivalence. One is the ‘hand-picked’ partition, while the second is the same as the three-cluster partition indicated by
the dendrogram in Figure 9.8. Similar results are obtained by using valued blockmodeling with \( m = 1 \). Valued blockmodeling with \( m \) equal to the minimal (but not zero) tie value produces the same complete block inconsistencies (if divided by \( m \)) as binary blockmodeling (with \( t \) equal to the minimal (but not zero) tie value). However, the null block inconsistencies are larger if some values in the block are larger than the minimal (nonzero) tie value. Therefore, it produces similar partitions as binary blockmodeling while assuring that the tie values in the null block (if any) are small. Consequently, the complete blocks may have more zero cells than they would in if binary blockmodeling would be used. This is often desired since it is often more desirable to have clear null blocks than clear complete blocks. This preference has often been expressed by giving larger weights to null block type inconsistencies. This is similar in some sense. However, here it is not the whole null block inconsistency that is given larger weight but the inconsistencies caused by individual ties that are weighted by the tie values of those ties. When applied to this network, value blockmodeling with \( m = 1 \) leads to only one optimal partition, namely the ‘hand-picked’ partition.

9.3.3 **Homogeneity and implicit blockmodeling – regular equivalence**

When performing blockmodeling according to regular equivalence, all six versions of homogeneity and implicit blockmodeling provide reasonable three-cluster partitions. All unique partitions (four)\(^83\) are presented on Figure 9.11. The six versions correspond to sum of squares and absolute deviations blockmodeling using \textit{max}-regular and \textit{mean}-regular equivalence (four versions) and implicit blockmodeling using either only regular blocks or null and regular blocks (two versions, in both cases a \textit{max}-regular partition). Binary and valued blockmodeling were also tested. When using \( t \) equal to 1, binary blockmodeling finds 238 of the optimal partitions. The use of larger values of \( t \) is not suggested since, as we noted before, all ties even those with a value of 1, are relevant. If larger values are nevertheless used this does not lead to reasonable partitions. Similar can be said for valued blockmodeling.

Absolute deviations with \textit{max}-regular equivalence and implicit blockmodeling with only regular blocks allowed produced the ‘hand-picked partition’. The partition obtained using implicit blockmodeling and null and regular blocks and absolute deviations is similar to the ‘hand-picked’ one, except that the units ‘SWC’ and ‘Disability commission’ are exchanged. In this case, the blockmodel\(^84\) (with null and regular blocks) by implicit blockmodeling is sensible, however this is only a coincidence\(^85\). The sum of squares blockmodeling according to

---

\(^83\) Some of these approaches produce the same partitions and are therefore presented on the same matrices in Figure 9.

\(^84\) The blockmodel is again indicated by shades instead of solid color indicating the tie strength in the null blocks on the lower left matrix in Figure 9.. The classification applies only to the implicit blockmodeling partition although the partition is also the same for absolute deviations blockmodeling and is therefore depicted in the same matrix.

\(^85\) ‘Sensible’ is used in the sense that regular blocks should have stronger ties (more precisely, on average larger row and column maximums) than the null blocks. It is always true with implicit blockmodeling with null and
max-regular equivalence and the absolute deviations blockmodeling according to mean-regular equivalence produced two similar partitions that differ quite noticeably from the ‘hand-picked’ partition.

**Figure 9.11:** Matrices partitioned using homogeneity and implicit blockmodeling according to regular equivalence

<table>
<thead>
<tr>
<th>Implicit b., regular blocks only</th>
<th>Absolute deviations b., max-regular</th>
<th>Sum of squares blockmodeling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDII</td>
<td>ESS</td>
<td>Health</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDII</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>ESS</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>Health</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>Municipality</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Schools</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>SWC</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>NGO</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Absolute deviations</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDII</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>ESS</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>Health</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>Municipality</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Med. of work</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SWC</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Psy., Addict.</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Schools</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Dis. comm.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NGO</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum of squares b., mean-regular</td>
<td>Implicit b., null and regular</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDII</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>ESS</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>Health</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>Med. of work</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SWC</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Psy., Addict.</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Municipality</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Schools</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Dis. comm.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NGO</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

We could say that the ‘hand-picked’ partition has a stronger emphasis on the pattern of the ties, while the rest are more focused on the strength of the ties. The rest should be interpreted slightly differently as the ‘hand-picked’ partition. The core cluster remains the same and is strongly connected internally. The second cluster is connected to the core cluster. In addition, regular blocks that the row and column maximums are on average closer to the block maximum than to 0 in regular blocks and closer to 0 in null blocks.
there are practically no other significant ties among the clusters. The difference between these two partitions is in which units are allocated to the second and which to the third (disconnected) cluster. In the third cluster there are either 2, 3 or 4 units with the lowest ties.

Again we can compute the values of the ties of the reduced graph. But since we are now using regular equivalence, the values of ties should be computed accordingly. However, the suitable value also depends on the way the f-regular equivalence is defined and, of course, on function f. This means that the appropriate way of computing the tie values depends on function f and the approach used. For more details, see Section 7.4.

Following the suggestions given there for absolute deviations with max-regular equivalence (which produced the ‘hand-picked’ partition), we can compute the values of the ties in the reduced graph by computing the mean of the two medians, one computed on the row and one on the column maximums. We could do something similar for the sum of squares blockmodeling by replacing the medians with the mean. The results of both versions are presented below for the ‘hand-picked’ partition, although this partition is not optimal for the sum of squares approach with max-regular equivalence:

<table>
<thead>
<tr>
<th>Absolute deviations</th>
<th>Sum of squares</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[1,] [2,] [3,]</td>
</tr>
<tr>
<td>[1,]</td>
<td>11 3 3</td>
</tr>
<tr>
<td>[2,]</td>
<td>3 1 0</td>
</tr>
<tr>
<td>[3,]</td>
<td>3 0 2</td>
</tr>
</tbody>
</table>

The same way used for computing the tie values of the reduced graph can also be considered as a ‘general’ way of computing the reduced values of reduced graphs as the mean is generally used to compute the representative value of a set of numbers. Other options are discussed in Section 7.4, however most of them lead to similar results and conclusions based on them.

The exception is the rule for implicit blockmodeling with null blocks. If we use implicit blockmodeling with null and regular blocks to obtain an image based on the ‘hand-picked’ partition we would obtain the following image and a valued reduced graph:

<table>
<thead>
<tr>
<th>Image</th>
<th>Valued reduced graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>&quot;reg&quot; &quot;null&quot; &quot;null&quot;</td>
</tr>
<tr>
<td>[2,]</td>
<td>&quot;null&quot; &quot;reg&quot; &quot;null&quot;</td>
</tr>
<tr>
<td>[3,]</td>
<td>&quot;null&quot; &quot;null&quot; &quot;reg&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>[1,] [2,] [3,]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>11 0 0</td>
</tr>
<tr>
<td>[2,]</td>
<td>0 1 0</td>
</tr>
<tr>
<td>[3,]</td>
<td>0 0 2</td>
</tr>
</tbody>
</table>

This again indicates that in most cases it is inadvisable to use implicit blockmodeling with null blocks. All computed valued reduced graphs presented in this subsection are presented in a graphic form in Figure 9.12.
Figure 9.12: Image graphs according to the max-regular equivalence for a ‘hand-picked’ partition

9.3.4 Conclusion

Most methods gave good partitions, although not for all numbers of clusters (for example, homogeneity blockmodeling according to structural equivalence did not find a good three-cluster partition). A lot of them came relatively close to the ‘hand-picked’ partition. They were especially good at recovering the core cluster. Based on the results, especially considering the regular equivalence, we see that the partitioning of the remaining units can be done in two ways: primarily based on the patterns of relations or primarily based on the strength of relations.

Two units were especially hard to classify – the ‘Disability Commission’ and ‘SWC’. The ‘Disability Commission’ really does not fit into any of the three clusters as it is the only unit without ties to any of the core units. On the other hand, ‘SWC’ could at least partially fit into all clusters based on the network data and from a substantive point of view.

From the network point of view, ‘SWC’ has relatively strong connections to the core (stronger than any other non-core unit) pooling it closer to the core, although merely based on the network data it does not really fit there. The strength of its ties to the core also makes the ‘SWC’ more similar to the other healthcare organizations (‘Medicine of work’, ‘Psychiatric organizations and organizations for helping with addiction’) than to ‘social environment’ organizations. The pattern of relations, on the other hand, clearly places the ‘SWC’ in the ‘social environment’ cluster, although the strength of its relations to the core is higher than that of the other organizations in the cluster.

‘SWC’ is also hard to place in only one of the clusters from the substantive perspective as they play multiple roles. On one hand, ‘SWC’ are authorized by the state to provide certain services such as monetary (and other) social support, which makes them similar to the ‘ESS’ and ‘PDII’ from the core cluster. A social worker might cooperate with them to obtain help for employees about to be fired or retired. On the other hand, they offer counseling and similar support, similarly to the ‘Psychiatric organizations and organizations for helping with addiction’ and ‘NGO’, where an employee can be sent for professional counseling. Further, they are also an important ‘community player’, as are the municipalities, schools, and certain health organizations. However, there is another important reason occupational social workers
might cooperate with ‘SWC’. Occupational social workers usually do not have many (if any) colleagues with a similar education at their workplace. When facing a professional dilemma, they often seek advice from a fellow social worker and many of these are employed at ‘SWC’. This might pull the ‘SWC’ closer to the core.

### 9.4 Example 4: Borrowing notes between social-informatics students

The data in this example come from a survey conducted in May 1993 on 13 social-informatics students (Hlebec, 1996). The network was constructed from answers to the question: ‘How often did you borrow notes from this person?’ for each of their fellow students. This network is presented in Figure 9.13 in matrix and graphic forms.

**Figure 9.13: Valued network of borrowing notes between social-informatics students**

The aim of the analysis is to discover clusters of students who have similar ties to the students of their and the other clusters. All types of generalized blockmodeling discussed in the paper are considered and compared.

It seems unlikely that a student would borrow notes from each student in a given cluster. Therefore, the concept of \((f-)\)regular equivalence seems more suitable. Several functions are used as \(f\). These functions are maximum and sum for valued blockmodeling and maximum and mean for homogeneity blockmodeling. The use of the maximum is based on the ideas of Batagelj and Ferligoj (2000: 12-13) and the REGGE algorithm (White, 1985a). There the maximum is presented as a suitable function for the generalization of regular equivalence to the valued networks. The function sum is also used in valued blockmodeling since it is assumed that students of a given cluster want to borrow a certain amount of notes from

---

86 The respondents indicated the frequency of borrowing by choosing (on a computer) a line of length 1 to 20, where 1 meant no borrowing. 1 was deducted from all answers so that 0 now means no borrowing.
students of another (or the same) cluster. The problem with this assumption is that the values do not represent the amount of notes borrowed. The function \textit{mean} used in homogeneity blockmodeling is equivalent to the function \textit{sum} used in valued blockmodeling.

Although regular equivalence is a more reasonable concept, methods for structural equivalence were nevertheless tested (although not in such depth). Unlike when using regular equivalence, a local search was used when looking for the optimal partitions using generalized blockmodeling approaches with 20 random starts. The reason for this lies in the better convergence of the algorithm when using structural equivalence and in the greater importance that is attributed to regular equivalence.

By inspecting the several partitions produced by the different types of generalized blockmodeling, allowed ideal blocks and with different numbers of clusters, it can be seen that a partition into three clusters is the most appropriate. Therefore, only partitions into three clusters are presented.

9.4.1 Indirect approach using structural equivalence

The analysis started with historically the first and also the simplest equivalence – structural equivalence. First the indirect approach was applied. Figure 9.14 shows the dendrogram obtained by Ward’s hierarchical clustering on corrected Euclidean-like dissimilarity (Burt and Minor, 1983 in Batagelj et al., 1992: 71) with \( p = 2 \).

Figure 9.14: Dendrogram obtained by Ward's hierarchical clustering on corrected Euclidean-like dissimilarity (Burt and Minor, 1983 in Batagelj et al., 1992: 71) with \( p = 2 \)

In Figure 9.15, the split into two clusters is very clear, but the remaining splits are not. The matrix partitioned according to a five-cluster partition is presented in Figure 9.15. The wider lines indicate splits higher in the dendrogram so it is possible to imagine the partition into fewer clusters. As we can see, none of the partitions seems very reasonable.
9.4.2 Direct approaches using structural equivalence

The homogeneity partitions presented in Figure 9.16 provide a slight improvement compared to the partition provided by the indirect approach.

The partitions obtained using implicit blockmodeling are presented in Figure 9.17. When using null and complete blocks, the blockmodel obtained is troublesome at best. The partition also seems more reasonable only when complete blocks are used.
The results of valued blockmodeling with parameter $m$ set to 1 and 3 are presented in the first two matrices in Figure 9.18. Higher values of parameter $m$ do not provide satisfactory results. Using slicing threshold $t$ equal to one binary blockmodeling finds four ‘optimal’ partitions. These are presented in matrices 2 to 5 (counting by rows first) in Figure 9.18. One of these partitions is the same as the one returned by valued blockmodeling with $m$ equal to 3. One of them is also the same as one of the two ‘optimal’ partitions obtained using binary blockmodeling with slicing threshold $t$ equal to 2. These two partitions are presented in the last two matrices in Figure 9.18. Slicing thresholds $t$ higher than 2 produced inferior partitions, namely partitions with most of the units concentrated in one cluster.

Especially the partitions obtained by the valued and the binary blockmodeling seem very reasonable and, as can be seen in the remainder of the example, are as good as those (some are even the same) obtained using regular equivalence.

### 9.4.3 Homogeneity blockmodeling according to regular equivalence

The network of borrowing notes is, according to regular equivalence, first analyzed using homogeneity blockmodeling. The optimal partition for sum of squares and absolute deviations blockmodeling\(^87\) according to mean-regular equivalence is presented in Figure 9.19.

---

\(^87\) In this case, both types of homogeneity blockmodeling produce the same partition.
Figure 9.18: Matrix partitioned using valued and binary blockmodeling according to structural equivalence

Figure 9.19: Optimal partitions for sum of squares and absolute deviations homogeneity blockmodeling according to mean-regular equivalence

Based on this partition, an appropriate value of parameter $m$ for valued blockmodeling according to sum-regular equivalence could be around 5 or 10. This assumption is based on the following two matrices that represent the mean row and columns sums in each block of that blockmodel.
However, instead of the function *mean* over rows and columns, the *maximum* could be also used. Or to put it differently, instead of searching for *mean*-regular blocks, *max*-regular blocks could be searched for. The optimal partitions for *max*-regular blocks are presented in Figure 9.20.

Figure 9.20: Optimal partitions for homogeneity blockmodeling according to *max*-regular equivalence

<table>
<thead>
<tr>
<th>Sum of squares blockmodeling</th>
<th>Absolute deviations blockmodeling</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Sum of squares blockmodeling" /></td>
<td><img src="image2" alt="Absolute deviations blockmodeling" /></td>
</tr>
</tbody>
</table>

Here the sum of squares and absolute deviations partitions do not match. A subjective judgment is needed to determine which partition is better. The absolute deviations partition induces blocks that seem ‘cleaner’. The upper and lower left blocks can now be more easily interpreted as null blocks and unit 8 fits quite nicely into the third cluster. Based on this partition, the appropriate $m$ for valued blockmodeling according to *max*-regular equivalence would be around 5. This assumption is based on the following two matrices that represent the mean row and columns maximums in each block of that blockmodel.

---

88 The procedure identifies these blocks as $f$-regular blocks. As noted in Section 5, null blocks are only a special case of $f$-regular blocks whereby the value of function $f$ for all rows and all columns is exactly 0. This rarely happens. However, when we are interpreting the result of the blockmodel we can interpret the $f$-regular blocks that are close to null blocks as null blocks.
Mean row maximums:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.8</td>
<td>5.2</td>
<td>16.0</td>
</tr>
<tr>
<td>2</td>
<td>8.4</td>
<td>2.0</td>
<td>9.0</td>
</tr>
<tr>
<td>3</td>
<td>2.0</td>
<td>0.0</td>
<td>14.7</td>
</tr>
</tbody>
</table>

Mean column maximums:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.8</td>
<td>5.4</td>
<td>17.0</td>
</tr>
<tr>
<td>2</td>
<td>7.8</td>
<td>2.6</td>
<td>12.7</td>
</tr>
<tr>
<td>3</td>
<td>2.0</td>
<td>0.0</td>
<td>13.0</td>
</tr>
</tbody>
</table>

9.4.4 Implicit blockmodeling according to regular equivalence

The results of two versions of implicit blockmodeling are presented in Figure 9.21, one with allowed ideal blocks null and regular, and the other with regular ideal blocks only. Where the approach produces an image (implicit blockmodeling using null and regular blocks, binary, and valued blockmodeling), the null blocks are indicated on the matrix by the shades of lines instead of shades of the solid color indicating the tie strength. These shaded lines are hard to see when the tie values are low. However, this itself indicates null blocks. Therefore, we can conclude from the first matrix in Figure 9.21 that the corresponding image is as follows:

```
1 2 3
1 "null" "null" "reg"
2 "null" "null" "null"
3 "null" "null" "null"
```

Figure 9.21: Networks of borrowing notes partitioned using implicit blockmodeling according to regular equivalence

The partition obtained using null and regular blocks is acceptable. It is, for example, the same as that obtained using implicit blockmodeling with only regular blocks or those obtained using absolute deviations with mean- of max-regular blocks. However, only one block is classified as regular. The partition obtained using only regular blocks is even better, mainly since unit 2 does not really belong to the third cluster (as ordered in Figure 9.21). It is similar to the absolute deviations partition; however blocks $R(1,2)$, $R(2,2)$ and $R(3,1)$ appear cleaner.
On the other hand, the additional member of the third cluster does not exactly fit. The interpretation nevertheless stays the same.

9.4.5 Valued blockmodeling according to regular equivalence

The best results for valued blockmodeling using null and sum-regular blocks were obtained with one of the suggested \( m \) values - 10. It produced the partition and the model presented in Figure 9.22. For valued blockmodeling according to max-regular equivalence, \( m \) equal to 5 is appropriate, again as suggested by the max-regular absolute deviations partition. This partition is identical to that obtained using sum-regular equivalence and therefore presented in Figure 9.22. The model is also the same (except the solid color now indicates max-regular blocks instead of sum-regular blocks). The same setting also produced a second, very similar partition with the same model. The only difference is that unit 13 moved from the first to the second cluster.

Figure 9.22: Optimal partition for valued blockmodeling obtained using null and max-regular blocks with \( m \) equal to 5 or using null and sum-regular blocks with \( m \) equal to 10

9.4.6 Binary blockmodeling according to regular equivalence

In this case binary blockmodeling does not produce as good results as the previous approaches. Binary blockmodeling is applied to the network in Figure 9.13 sliced at a threshold \( t \) equal to 1, 2, 3, 5 and 10 (values of ties equal or greater to these values were recoded into ones). These values were suggested by the histogram in Figure 9.23.

89 Again based on a subjective judgment.

90 The model is indicated by the pattern in the non-null cells of a given block. Solid color indicates regular (in this case sum-regular blocks) while the shaded lines indicate null blocks.
Figure 9.23: Histogram of cell values for the matrix in Figure 9.13

All of the three-cluster partitions contain obvious misclassifications\textsuperscript{91}. The best of them are presented in Figure 9.24. The first solution is based on the network sliced at the value of 1. It is presented in the first matrix in Figure 9.24 as a binary network and in the second matrix as a valued network. The biggest problem with this solution is that unit 4 is not in the first cluster. This is not so evident if we look at the binary network, but with the additional data the valued network provides it is obvious. The same setting (the matrix in Figure 9.13 sliced at threshold 1) produces another optimal solution, which is presented in the third matrix in Figure 9.24.

The partitions presented in the third (the same partition as obtained in the network sliced at threshold 1 and is therefore presented in the same matrix) and fourth matrices in Figure 9.24 were obtained in the network sliced at 2. The partition of the second solution is the same as that of the second solution obtained in the network sliced at 1. The solutions obtained in the network sliced at 2 are reasonably good although it is obvious that at least unit 8 in the first solution and unit 4 in the second solution are misclassified. In the fifth matrix in Figure 9.24 the solution obtained in the network sliced at the value of 3 is presented. As in the previous partitions, at least one unit (unit 4) is misclassified. A search for a three-cluster solution in a network sliced at 5 and 10 does not produce satisfactory results. E.g., for slicing threshold $t$ equal to 5 it finds 100 partitions with inconsistencies equal to the minimum of the inconsistencies of all partitions.

\textsuperscript{91} The two-cluster solution has other deficiencies.
Figure 9.24: Optimal partitions for binary blockmodeling according to regular equivalence with different values of slicing threshold $t$

9.4.7 REGE

Most versions of REGE presented in Section 4.4 are also applied to this network. The algorithms run for ten interactions so that the differences between subsequent iterations are small. As the output of all versions is a (dis)similarity matrix, a clustering algorithm is needed to find a partition. The choice of an appropriate clustering algorithm is essential. In the similarity matrix obtained with REGE, three hierarchical clustering methods (Ward's method, complete linkage, and single linkage) were tested to demonstrate the effect of a clustering method. Their dendrograms are presented in Figure 9.25. Ward's method and the complete linkage produce the same partition at three (and four) clusters, while the three-cluster single linkage partition is profoundly different and does not partition the matrix into the blocks well. Ward's method and the complete linkage produced the same partitions in about half of the cases; however, only the partitions produced by Ward's method are presented (since they gave superior results).

The first matrix in Figure 9.26 presents the partition produced by REGGE, REGGE-OW, and REGDI-OW by Ward's or complete linkage hierarchical clustering.

The other three algorithms give different partitions. The partitions obtained using these algorithms also differed depending on the hierarchical clustering method used (Ward's or complete linkage, only partitions obtained by Ward's method are presented). These partitions are presented in the remaining three matrices in Figure 9.26.
The partition obtained using REGGE, REGGE-OW and REGDI-OW puts unit 2 in the third cluster, where it does not belong. Block R(1, 1) based on the partition produced by REGDI is...
very inhomogeneous. The partition obtained using REGGE-OCOW puts unit 8 in the second cluster, although it probably belongs more to the third cluster. Due to these shortcomings of other partitions, the partition obtained using REGGE-OC seems to be the best partition out of all the REGE partitions. It is very similar to the max-regular and mean-regular absolute deviations partitions and implicit blockmodeling (with only regular blocks).

9.4.8 Interpretation

In this example we have seen that several methods can produce similar or even the same partitions. In spite of this, a large number of different partitions were produced by all the methods considered. It is hard to select the most appropriate partition out of those presented since they were obtained using different equivalences, different criterion functions and different definitions of ideal blocks. A researcher can select a partition based on several criteria, some of which are:

- the suitability of the (definition of) equivalence (which can also be defined through ideal blocks and optionally a blockmodel) used;
- the possibility of a logical interpretation of the partition and obtained blockmodel; and
- a subjective judgment of how well the empirical block corresponds to the ideal ones.

The third point would ideally be incorporated into the first one (definitions of ideal blocks) or more generally into the criterion function. However, this point is added as some researcher preferences are hard to include in the specification of the criterion function.

As mentioned at the beginning, regular equivalence is more suitable in this context. The question of which version of it (which definition of ideal blocks) is the most suitable is however hard to answer. If the network represented the actual flows of notes the functions sum and mean would be the most suitable, while in this case the maximum is probably at least as suitable as the mean and sum, if not more so.

On this basis the partitions obtained using structural equivalence are discarded, although some are very good. For example, the partition that is possibly the best partition is obtained using valued blockmodeling according to either max-regular equivalence at \( m = 5 \) or sum-regular equivalence at \( m = 10 \). However, the same partition is also obtained with valued blockmodeling at \( m = 3 \) and binary blockmodeling at \( t = 1 \) (as one of four ‘optimal’ partitions) according to structural equivalence.

Of the partitions based on \( f \)-regular equivalence the partitions that seem the most suitable are the valued blockmodeling partition (obtained with a model allowing null blocks and either max-regular blocks with \( m = 5 \) or sum-regular blocks with \( m = 10 \)), both absolute deviations blockmodeling partitions (with max- and mean-regular blocks) and the implicit blockmodeling partition with only regular blocks as allowed ideal blocks.

In all of these partitions, a cluster exists from which everyone borrows notes and whose members do not borrow notes from anybody outside their cluster. This cluster consists of units 4, 8, 9 and sometimes unit 3. These are probably the students who have good notes on a regular basis (presumably good students).
The interpretation of the other two clusters differs depending upon which of the solutions we make the interpretation. Based on the partitions obtained using absolute deviations blockmodeling according to either mean- or max-regular equivalence, implicit blockmodeling and REGGE-OC, the other two clusters exhibit similar behavior. They borrow notes from both remaining clusters (besides themselves) and borrow notes from each other; however, they only seldom borrow notes from within their cluster. The main difference between these two clusters is that one of them relies more heavily on the first cluster (good students). This might be two clusters of students that do not have much contact with the members of their cluster. This lack of within-cluster exchange may also indicate that the partitions obtained are inappropriate.

The interpretation of the valued blockmodeling solution seems more logical. Again, there are two clusters in addition to the ‘good students’ cluster. Both of them borrow notes from the good students’ cluster, while only one of them also borrows notes within its own cluster. They rarely borrow from each other. One of them could be a cluster of students who do not have much contact with fellow students and the other could be the ‘average’ students.

Other partitions (sum of squares blockmodeling according to max-regular equivalence, binary blockmodeling, and other versions of REGE) seem to be quite good at finding the main lender cluster, but not as good at partitioning the remaining units.

**9.5 Example 5: A ‘sweet’ island from EAT (the Edinburgh Associative Thesaurus)**

This data represent an island (called ‘sweet’) from the Edinburgh Associative Thesaurus (EAT) [http://www.eat.rl.ac.uk/]. In the network based on EAT the units represent the words in the thesaurus. The ties represent associations and go from the word used as a stimulus to the word given as a response. The tie value is the number of people who reported that response to a given stimulus. Each word that was used as a stimulus was given as such to 100 people.

As the whole network is too large to be analyzed with generalized blockmodeling, an ‘island’ was extracted from it. The island refers to the line island, a connected small subnetwork with stronger internal cohesion (higher tie values) relative to its neighborhood (Zaveršnik, 2003; Batagelj and Zaveršnik, 2004). The unordered matrices (valued and binary) representing the network are presented in Figure 9.27.

This network is presented here as an example of a larger network where a full search is not feasible. Further, the network is also measured on the ratio (and therefore also the interval) level of measurement. Another nice property of this network is that here the difference between 0 and 1 can be given approximately the same importance as the difference e.g. between 1 and 2. Due to the distribution of tie values the network is also suitable for most of the generalized blockmodeling approaches presented in this dissertation. The only exception is implicit blockmodeling (with the null block type), which is less appropriate due to the
presence of the right tail in the distribution. For all these reasons, this network was chosen as an example.

Figure 9.27: Matrix and graphic representation of the ‘sweet’ island from EAT

9.5.1 Structural equivalence – indirect approach

We observe that this network is relatively sparse and it seems that structural equivalence is not a very useful concept for finding larger clusters in this network. To check this quickly we can use an indirect approach to find structural equivalence clusters by first computing dissimilarities in terms of structural equivalence and then using them in hierarchical clustering. The dendrogram for Ward's method (Ward, 1963) is shown in Figure 9.28. It does not exhibit any clear clustering, as was expected after looking at the matrix of the network.

Figure 9.28: Dendrogram obtained by Ward's hierarchical clustering on dissimilarities in terms of structural equivalence computed from the network of the ‘sweet’ island
9.5.2 REGE

Regular equivalence is more appropriate for sparse networks. Since REGE algorithms are much faster and also do not need the number of clusters to be specified in advance, they are very suitable for gaining a first impression about the appropriateness of the regular equivalence. The dendrograms for the REGGE-OW and REGDI-OW\textsuperscript{92} algorithms (with ten iterations) are shown in Figure 9.29. They both exhibit a relatively clear partition in either two or three clusters.

*Figure 9.29: Dendrograms obtained by Ward's hierarchical clustering on (dis)similarities in terms of regular equivalence computed from the network of the ‘sweet’ island*

The partitioned matrices can be seen in Figure 9.30. The blue line indicates the split into two clusters, while the black line indicates the additional split into a three-cluster partition. Although they are by no means perfect solutions, the partition into three clusters obtained using REGGE-OW seems quite reasonable. Although REGE does not classify blocks into regular or null, we could classify especially those from the three-cluster model based on the partitioned matrix as follows:

```
1 2 3
1 "null" "null" "reg"
2 "null" "reg" "null"
3 "null" "null" "null"
```

With this we conclude the preliminary analysis of the network using indirect approaches and move to generalized blockmodeling approaches. Based on these results two- and three-cluster partitions are considered in the following subsections.

\textsuperscript{92} The original versions of the algorithms produced very similar dendrograms and the same partitions into two clusters.
9.5.3 Binary blockmodeling according to regular equivalence

First, binary blockmodeling as the historically first generalized blockmodeling approach was used. Also, it is the only one implemented in Pajek 1.11 (Batagelj and Mrvar, 2006). The implementation in Pajek is much faster and therefore it is possible to use more (random) starting partitions. When using binary blockmodeling, a local search with at least 500 starting partitions was used, while for other blockmodeling approaches usually only about 20 random starting partitions were used in addition to the partitions that were found to be optimal using other approaches (for example binary blockmodeling).

For the binary approach, the ‘optimal’ partition is one large regular block and some ‘marginal’ null blocks. The partition is presented in Figure 9.31. The resulting model is

```
1      2
1 "null" "null"
2 "null" "reg"
```

The results using three clusters are very similar and not very helpful as the additional split occurs in the small cluster.

---

93 The blockmodel can also be seen in Figure 9.31 – the squares representing the ties in the matrix are shaded (and not solid) in the null blocks.
The results of the binary approach can be improved by first selecting a threshold \((t)\) so that the ties with values under the threshold are treated as 0. When the threshold is set to just 2, the improvement is significant and quite a good solution is obtained. The partition is presented in the first matrix Figure 9.33. The appropriate model is:

1 2
1 "null" "reg"
2 "null" "null"

However, the selection of the appropriate threshold should not involve guesswork. Some useful information can be obtained by looking at the histogram of the tie values (0s are excluded). This histogram is presented in Figure 9.32. A rule of thumb here would be that the appropriate threshold is somewhere around where the density is minimal (for the first time). In our case, this would be somewhere around 20. Nevertheless, several other threshold values were tested.

The partition for thresholds 3, 4, 5 and 10 is presented in the second matrix on Figure 9.33. The solution is then the same for the threshold from 5 to 30. For thresholds \((t\) values) 5 and 10 more than one partition is ‘optimal’ and that is why the ranges of \(t\) values for the last two matrices overlap. These both are presented in the second and third matrices in Figure 9.33. They are all very similar to the partition at threshold 2 and share the same model. At a threshold of 40 and over no meaningful result was obtained. We see that the threshold 20, as suggested by the ‘rule of thumb’, also produces satisfactory results. All the solutions reveal we have a cluster of words that generate and a cluster of words that receive associations.
Figure 9.32: Histogram of tie values (0s are excluded) in the network of the ‘sweet’ island

![Histogram of tie values](image)

Figure 9.33: ‘Optimal’ partitions into two clusters of the ‘sweet’ island obtained with binary blockmodeling according to regular equivalence for networks sliced at different thresholds (t)

![Partition diagrams](image)

Three clusters partitions were also searched for. Here, each different value of slicing parameter t resulted in a different partition (in some cases several ‘optimal’ partitions were also found). As a result, too many partitions were found for all of them to be displayed. In Table 9.1, the values of the total inconsistencies (values of the criterion function) of the ‘optimal’ partitions into two and three clusters at different values of the slicing parameter t are presented. We can see that there are no improvements in the total inconsistency of partitions when we increase the number of clusters from two to three 3 when the value of the slicing threshold t is 1, 10 and 20. At slicing thresholds t 2 to 5 the improvements are small. However, interesting alternative partitions are obtained at slicing thresholds t 30 and 40 the improvements are considerable. The partitions obtained using slicing thresholds t equal to 3, 5
(one of two ‘optimal’) and 30 are presented in Figure 9.34. We note that the blockmodel is different for each solution. All blockmodels indicate that, similarly as in the two-cluster solutions, there is a cluster that generates and a cluster that receives most associations. The third cluster does not have any such stable characteristics. In the solution obtained using the slicing threshold \( t \) equal to 30, the abovementioned tie between the cluster that generates and the cluster that receives the most associations is the only tie between the clusters. In the other two cases there is one additional tie. In the solution obtained using slicing threshold \( t \) equal to 3 the remaining cluster is tied to itself. The same is also true for one of the solutions obtained using slicing threshold \( t \) equal to 5 (not depicted in Figure 9.34). In the other solutions obtained using slicing threshold \( t \) equal to 5 (presented in the second matrix in Figure 9.34) the remaining cluster is isolated, while the cluster receiving most associations is also receiving associations from its own cluster. Partitions into more than three clusters are not explored here in order to limit the length of this example.

Table 9.1: Total inconsistencies of ‘optimal’ partitions into two and three clusters obtained with binary blockmodeling according to regular equivalence

<table>
<thead>
<tr>
<th>( t )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k = 2 )</td>
<td>4</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>6</td>
<td>4</td>
<td>15</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>( k = 3 )</td>
<td>4</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 9.34: ‘Optimal’ partitions into three clusters of the ‘sweet’ island obtained with binary blockmodeling according to regular equivalence for networks sliced at different thresholds (\( t \))

9.5.4 Valued blockmodeling according to \( \text{max} \)-regular equivalence

The next approach to employ is valued blockmodeling according to \( \text{max} \)-regular equivalence. For this approach, parameter \( m \) must be selected in advance. A rule of thumb suggests that the value of \( m \) should be approximately twice the value of the threshold suitable for binary blockmodeling. This indicates that appropriate values should be searched for in the range

---

94 The blockmodel is indicated by the use of shaded lines instead of solid color to indicate the ties in the null blocks.
from 5 to 60. Several values on this interval were checked. The most reasonable partitions are obtained when \( m \) is between 10 and 40. Larger values produced less meaningful partitions (only two units in one cluster). The partitioned matrices can be seen in Figure 9.35. The obtained model can also be seen from these matrices as the null blocks are indicated by the shaded\(^95\) (not solid) squares indicating the cell values. The model is different for lower values of parameter \( m \) (1 to 5) than that obtained with binary blockmodeling. In these cases, the cluster that is the main receiver of associations also receives associations from its own cluster.

Figure 9.35: Matrices of the network of the ‘sweet’ island partitioned into two clusters using valued blockmodeling according to max-regular equivalence

\[^95\] The shading is very hard to notice for low cell values.
Partitions into three clusters were also searched for. The total inconsistencies for the two- and three-cluster partitions are presented in Table 9.2. Interestingly, the total inconsistency rises (when we increase the number of clusters from 2 to 3) for values of parameter $m$ 1 to 3 and 20 to 30. The most noticeable decreases in inconsistencies are observed at parameter $m$ values 5, 40 and 50. As was the case when using binary blockmodeling, each value (of those tested) of parameter $m$ produced a different partition. As there are consequently too many partitions for all to be presented here, mostly those where the largest decrease in total inconsistency is observed (where $m$ with values of 5, 40 and 50 was used) are presented in Figure 9.36. In addition, the partition at $m$ value 10 is also presented which provides a very reasonable partition.

Table 9.2: Total inconsistencies of 'optimal' partitions into two and three clusters obtained with valued blockmodeling according to max-regular equivalence

<table>
<thead>
<tr>
<th>$m$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k = 2$</td>
<td>25</td>
<td>64</td>
<td>111</td>
<td>158</td>
<td>198</td>
<td>218</td>
<td>218</td>
<td>242</td>
<td>416</td>
<td>861</td>
</tr>
<tr>
<td>$k = 3$</td>
<td>27</td>
<td>75</td>
<td>118</td>
<td>148</td>
<td>162</td>
<td>211</td>
<td>234</td>
<td>251</td>
<td>309</td>
<td>579</td>
</tr>
</tbody>
</table>

9.5.5 Valued blockmodeling according to sum-regular equivalence

For valued blockmodeling according to sum-regular equivalence, the process of selecting the correct value of parameter $m$ is even more complicated. The procedure suggested in Subsection 7.1.1 is used to determine possible $m$ values. In this example sum-regular blocks are used (the described procedure is also useful when row-sum-regular or column-sum-regular blocks are used). Therefore, the distributions of row and column sums (their means or two separate distributions) should be looked at to determine a suitable value of parameter $m$.

If we are looking at the distribution of row sums (in the first histogram in Figure 9.37), we can see that the distribution (if we ignore the very small values) slightly resembles the normal distribution. This does not give us a good clue about the value of parameter $m$. Parameter $m$ values from around 15 (or maybe even less) through to around 40 or 50 seem reasonable. The distribution even indicates that row-sum-regular blocks might not be the most appropriate. In the distribution of column sums (in the second histogram in Figure 9.37) $m$ values from 20 to about 40 seem appropriate. Looking at both of these two histograms together or looking at the third histogram in Figure 9.37, the best values of parameter $m$ when using valued blockmodeling according to sum-regular equivalence seem to be around 20. These estimates are made assuming that each cluster participates in at most one sum-regular block as a row cluster and in at most one sum-regular block as a column cluster. Based on the results for valued blockmodeling according to max-regular equivalence this is a realistic assumption for both the two- and three-cluster models.
Figure 9.36: Matrices of the network of the ‘sweet’ island partitioned into three clusters using valued blockmodeling according to max-regular equivalence

Figure 9.37: Distributions of the row sums, column sums and the means of the row and column sums
However, if we assume that at least some clusters participate in more sum-regular blocks as row or column clusters the values obtained based on the distribution of the row and column sums must be divided by the number of sum-regular blocks in which a cluster may participate as a row and column cluster, respectively. Some results of valued blockmodeling according to max-regular equivalence at lower values of parameter \( m \) do indicate the possibility of a cluster participating in two max-regular blocks as a column cluster. This would indicate that values of parameter \( m \) as low as 10 might be appropriate.

The obtained partitions into two clusters are the same as when using max-regular equivalence, although sometimes at different parameter \( m \) values. Table 9.3 shows which values of parameter \( m \) when using sum-regular equivalence resulted in the partitions shown in Figure 9.35 (partitions obtained according to max-regular equivalence). Therefore, the figures are not repeated here.

Table 9.3: The values of parameter \( m \) when using sum-regular equivalence that resulted in the max-regular partitions of the network of the ‘sweet’ island

<table>
<thead>
<tr>
<th>sum-regular equivalence</th>
<th>1 to 5</th>
<th>10 to 50</th>
<th>60 to 90</th>
</tr>
</thead>
<tbody>
<tr>
<td>max-regular equivalence</td>
<td>1 to 4</td>
<td>10 to 40</td>
<td>50 to 90</td>
</tr>
</tbody>
</table>

If we increase the number of clusters from two to three, the total inconsistency reduces at some values of parameter \( m \). This is shown in Table 9.4. The pattern here is similar as when using valued blockmodeling according to max-regular equivalence. The obtained partitions are mostly also the same as those obtained when using valued blockmodeling according to max-regular equivalence. The only differences occur at values of parameter \( m \) 4, 50 and 60. The one at value \( m \) equal to 4 is similar to the one at \( m \) equal to 3, while the one at \( m \) equal to 50 is similar to the one at \( m \) equal to 40. The partition obtained with valued blockmodeling according to sum-regular equivalence at \( m \) equal to 60 is the same as the partition obtained with valued blockmodeling according to max-regular equivalence at \( m \) equal to 50. Since the results are so similar to those obtained using valued blockmodeling according to max-regular equivalence they are not presented.

Table 9.4: Total inconsistencies of 'optimal' partitions into two and three clusters obtained with valued blockmodeling according to sum-regular equivalence

<table>
<thead>
<tr>
<th>( m )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k = 2 )</td>
<td>25</td>
<td>53</td>
<td>100</td>
<td>147</td>
<td>194</td>
<td>218</td>
<td>218</td>
<td>236</td>
<td>392</td>
<td>746</td>
<td>881</td>
</tr>
<tr>
<td>( k = 3 )</td>
<td>27</td>
<td>65</td>
<td>107</td>
<td>141</td>
<td>162</td>
<td>211</td>
<td>234</td>
<td>245</td>
<td>309</td>
<td>470</td>
<td>746</td>
</tr>
</tbody>
</table>

9.5.6 Homogeneity blockmodeling according to max- and mean-regular equivalence

Both homogeneity approaches were applied with max- and mean-regular equivalence. The partitions into two clusters for all four possible combinations are presented in Figure 9.38. All partitions are reasonable and relatively similar, however no two partitions are the same. As
expected, the choice of function $f$ used in $f$-regular equivalence (mean or max) seems to have a stronger effect on the partition obtained than the choice of the measure of variability.

Figure 9.38: Matrices of the network of the ‘sweet’ island partitioned into two clusters using homogeneity blockmodeling according to $f$-regular equivalence.
Table 9.5: Total inconsistencies of the 'optimal' partitions into two and three clusters obtained with homogeneity blockmodeling

<table>
<thead>
<tr>
<th>function $f$</th>
<th>maximum</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>blockmodeling type</td>
<td>sum of squares</td>
<td>absolute deviations</td>
</tr>
<tr>
<td>$k = 2$</td>
<td>50007.29</td>
<td>2608</td>
</tr>
<tr>
<td>$k = 3$</td>
<td>32246.67</td>
<td>1748</td>
</tr>
</tbody>
</table>

Figure 9.39: Matrices of the network of the 'sweet' island partitioned into three clusters using homogeneity blockmodeling according to $f$-regular equivalence

9.5.7 Implicit blockmodeling according to max-regular equivalence

The last approach is implicit blockmodeling. The results for two versions are presented in Figure 9.40, one with allowed null and regular block types and one with only regular block types. Formulas for the block inconsistencies presented in Table 7.7 were used. These are the formulas where neither block size normalizations nor maximum normalization is used. Normalized versions were also tested; however, they performed similarly or worse.

Here it is clearly seen that the explicit use of null blocks leads to an unsatisfactory solution. The reason for this is best understood if we compare the method with valued blockmodeling. The methods are very similar, with the main difference being that parameter $m$ in valued blockmodeling is replaced by the block maximum. Most of the blocks that could be close to regular have some very high values, which means that the block maximum is very high. As we can see when the network was analyzed by valued blockmodeling, this leads to very high block inconsistencies for regular blocks compared to null blocks. However, since in implicit blockmodeling regular blocks can also be used to measure the deviation from null blocks, this problem can be overcome by only using regular blocks.

96 See Section 7.3 (Implicit blockmodeling) for a justification.
Figure 9.40: Matrices of the network of the ‘sweet’ island partitioned into two clusters using implicit blockmodeling according to max-regular equivalence

![Matrix Diagram]

As with other approaches, whether an increase in the number of clusters (from two to three) reduces the total inconsistency was also explored. This is shown in Table 9.6. The obtained partitions are shown. Although the decrease in the total inconsistencies exhibited in Table 9.6 is relatively large, the results in terms of the partitioned matrices presented in Figure 9.41 do not yield any noticeable improvements.

Table 9.6: Total inconsistencies of the ‘optimal’ partitions into two and three clusters obtained with implicit blockmodeling

<table>
<thead>
<tr>
<th>blocks used</th>
<th>regular only</th>
<th>null and regular</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k = 2$</td>
<td>6615</td>
<td>855</td>
</tr>
<tr>
<td>$k = 3$</td>
<td>4271</td>
<td>715</td>
</tr>
</tbody>
</table>

9.5.8 Comparison and interpretation of the results

In this subsection, the two-cluster partitions obtained using generalized blockmodeling are compared using the Adjusted Rand Index (Hubert and Arabie, 1985: 198). The three-cluster partitions are not compared using the Adjusted Rand Index mainly since not all approaches produced good three-cluster partitions and because there are too many of them.

The matrix of Adjusted Rand Indices obtained in pair-wise comparisons of two-cluster partitions is presented in Figure 9.42. The values of the Adjusted Rand Indices presented in Figure 9.42 are multiplied by 10 and rounded off. The exact figures can be seen in Appendix C. The maximal value of the Adjusted rand index is 1, which indicates that two partitions are identical. The other extreme is not well defined. However, the Adjusted Rand Index is designed so that it has a value of 0 when comparing two partitions that are as similar as can be expected by chance (Hubert and Arabie, 1985: 198).
Figure 9.41: Matrices of the network of the 'sweet' island partitioned into three clusters using implicit blockmodeling according to max-regular equivalence

The blue lines separated the partitions obtained by different approaches and in some cases different options within approaches are divided by the black (thinner) lines. We see that, overall, the partitions are quite similar. Some are even identical. However, two partitions are completely different from the rest, one obtained with binary blockmodeling by treating all non-zero values as ties (bin,t1) and one obtained with both versions of valued blockmodeling at high values of m (val,max,m50-90; val,sum,m60-90) and with implicit blockmodeling with allowed block types null and regular (imp, null & reg).

Figure 9.42: Graphic representation of Adjusted Rand Indices obtained by comparing of two-cluster partitions
There are especially a lot of matches (identical partitions) between two versions of valued blockmodeling (one using max- and the other sum-regular equivalence). This is not a surprise as the network is very sparse and the maximum is mostly not much smaller than the sum. Also, identical partitions are obtained using valued blockmodeling according to max-regular equivalence at $m$ from 10 to 40 (Figure 9.35), valued blockmodeling according to sum-regular equivalence at $m$ from 10 to 50, and sum of squares blockmodeling according to max-regular equivalence (Figure 9.38).

This is also probably the best partition. If we look at its blockmodel (obtained with one of the versions of valued blockmodeling) the interpretation is also clear. There are two clusters of words inside this network. When used as a stimulus words from the larger cluster produce an association to one of the words of the second, smaller cluster. The words of the first cluster receive very few associations, at least from the words selected in this island (subnetwork). The second cluster consists of words that were often returned as a response (association).

Other reasonable partitions have similar interpretations. Some (for example, those obtained using valued blockmodeling (both versions) at lower $m$ values) indicate that the second (smaller) cluster is the main receiver of associations from both clusters of words. However, the regular blocks in such partitions are much less regular, with several columns having low values only.

While most blockmodeling types produce relatively good two-cluster partitions of the ‘sweet’ network, this cannot be said for the three-cluster partitions. Often, these do not justify the use of an additional cluster. Yet there are some exceptions. These are the three-cluster partitions obtained using binary and valued blockmodeling using low values of parameters $t$ and $m$. However, the intervals of the values of parameters $t$ and $m$ with which such partitions can be obtained are very narrow.

### 9.6 Example 6: St. Marks carbon flow data

In this section, the St. Marks carbon flow network (Baird et al., 1998) was analyzed. The network contains flows of carbon (measured in mg/m$^2$/day) between 51 different units measured in a large seagrass ecosystem in the St. Marks National Wildlife Refuge, in Apalachee Bay in the Gulf of Mexico, Florida.

#### 9.6.1 Previous analyses

The network was already analyzed with blockmodeling by Luczkovich et al. (2003) and Nordlund (2007). Luczkovich et al. (2003) used REGE to identify isotropic positions or equivalence classes. Nordlund (2007) did not search for a suitable partition. He noted that Luczkovich et al. (2003) used criteria whereby any non-zero value in a block resulted in a tie in a reduced graph. He presented an alternative rule for identifying regular blocks in valued blockmodeling.

---

97 The ‘units’ can be spices, groups of spices or even the line of substances.
networks. He took the partition obtained by Luczkovich et al. (2003) and applied a new rule for identifying the ties in the reduced graph. The rule that Nordlund (2007) developed in the paper was also an inspiration for the development of the new versions of REGE presented in Section 4.4.3 and for using generalized blockmodeling as suggested in Subsection 7.5.7.

Graphic representations of the network (without an indication of the tie values) as presented in Luczkovich et al. (2003) can be seen in Appendix D. The colors of the vertices indicate the partition obtained by Luczkovich et al. (2003). The Appendix D also contains a reduced graph based on this partition.

As mentioned above, Nordlund (2007) was only concerned with identifying regular blocks (distinguishing them from null blocks) for the purpose of obtaining a suitable image/reduced graph and not with finding a suitable partition. Therefore, he started with the partition obtained by Luczkovich et al. (2003) and tried to obtain a better reduced graph. He used a row-normalized network to assess if a certain block was row-regular and column-normalized network to assess if it is column-regular. He had done this after he binarized the normalized networks using the threshold (cutoff value) equal to the inverse of the number of actors (1/51). In Figure 9.43 the row and column normalized networks are presented. In these two matrices, cells with relatively low values (let us say 0.1) can be hardly seen, although Nordlund (2007) treats all values over approximately 0.02 as being equally important. For this, the binarized versions (with a threshold 1/51 ≈ 0.02) of these matrices are presented in Figure 9.44.

Nordlund (2007) computed for each block the number of rows matrix that have at least one tie in it in a binarized row normalized matrix plus the number of columns matrix that have at least one tie in it in a binarized column normalized matrix and then divided that by the total number of rows and columns in a block. If the proportion was greater than (or equal to) some selected proportion he declared the blocks regular. His blockmodel is also indicated in Figure 9.44; the ‘full’ cells in null blocks are dashed, not solid. He allowed that the statistics for rows and columns could be combined differently. This might be necessary as the classification of a block into regular or null with one dimension significantly larger than the other is mainly affected by the larger dimension. This is, for example, clearly seen in block $R(6, 2)$.

He produced a reduced graph presented in Appendix F in a similar way as the reduced graph proposed by Luczkovich et al. (2003) in Appendix E with much fewer ties compared to that produced by Luczkovich et al. (2003) due to the application of a more stringent rule for ties in the reduced graph, however, there are still 10 clusters which makes the reduction relatively small. Seven out of these 10 clusters have three or less units, and three of them just one unit? Surely, at least for some applications, a less detailed picture (reduced graph) with fewer units is desired.
Figure 9.43: Row and column normalized matrices of the St. Marks carbon flow network
### Figure 9.44: Binarized row and column normalized matrices of the St. Marks carbon flow network

#### Row normalized

```
<table>
<thead>
<tr>
<th>Zooplankton</th>
<th>Macro-epiphytes</th>
<th>Suspension-feed polychaetes</th>
<th>Detritus-feed crustaceans</th>
<th>Spider crab</th>
<th>Blue crab</th>
<th>Predatory shrimp</th>
<th>Detritus-feed polychaetes</th>
<th>Epiphyte-grazing gastropods</th>
<th>Other gastropods</th>
<th>Tongue fish</th>
<th>Gulf flound &amp; needlefish</th>
<th>Atlantic silverside &amp; bay anchovies</th>
<th>Pinfish</th>
<th>Spot</th>
<th>Pipefish &amp; seahorses</th>
<th>Sheepshead minnow</th>
<th>Red Drum</th>
<th>Southern hake &amp; sea robins</th>
<th>Gobies &amp; blennies</th>
<th>Killifish</th>
<th>Herbivorous ducks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zooplankton</td>
<td>Macro-epiphytes</td>
<td>Suspension-feed polychaetes</td>
<td>Detritus-feed crustaceans</td>
<td>Spider crab</td>
<td>Blue crab</td>
<td>Predatory shrimp</td>
<td>Detritus-feed polychaetes</td>
<td>Epiphyte-grazing gastropods</td>
<td>Other gastropods</td>
<td>Tongue fish</td>
<td>Gulf flound &amp; needlefish</td>
<td>Atlantic silverside &amp; bay anchovies</td>
<td>Pinfish</td>
<td>Spot</td>
<td>Pipefish &amp; seahorses</td>
<td>Sheepshead minnow</td>
<td>Red Drum</td>
<td>Southern hake &amp; sea robins</td>
<td>Gobies &amp; blennies</td>
<td>Killifish</td>
<td>Herbivorous ducks</td>
</tr>
<tr>
<td>Zooplankton</td>
<td>Macro-epiphytes</td>
<td>Suspension-feed polychaetes</td>
<td>Detritus-feed crustaceans</td>
<td>Spider crab</td>
<td>Blue crab</td>
<td>Predatory shrimp</td>
<td>Detritus-feed polychaetes</td>
<td>Epiphyte-grazing gastropods</td>
<td>Other gastropods</td>
<td>Tongue fish</td>
<td>Gulf flound &amp; needlefish</td>
<td>Atlantic silverside &amp; bay anchovies</td>
<td>Pinfish</td>
<td>Spot</td>
<td>Pipefish &amp; seahorses</td>
<td>Sheepshead minnow</td>
<td>Red Drum</td>
<td>Southern hake &amp; sea robins</td>
<td>Gobies &amp; blennies</td>
<td>Killifish</td>
<td>Herbivorous ducks</td>
</tr>
</tbody>
</table>
```

#### Column normalized

```
<table>
<thead>
<tr>
<th>Zooplankton</th>
<th>Macro-epiphytes</th>
<th>Suspension-feed polychaetes</th>
<th>Detritus-feed crustaceans</th>
<th>Spider crab</th>
<th>Blue crab</th>
<th>Predatory shrimp</th>
<th>Detritus-feed polychaetes</th>
<th>Epiphyte-grazing gastropods</th>
<th>Other gastropods</th>
<th>Tongue fish</th>
<th>Gulf flound &amp; needlefish</th>
<th>Atlantic silverside &amp; bay anchovies</th>
<th>Pinfish</th>
<th>Spot</th>
<th>Pipefish &amp; seahorses</th>
<th>Sheepshead minnow</th>
<th>Red Drum</th>
<th>Southern hake &amp; sea robins</th>
<th>Gobies &amp; blennies</th>
<th>Killifish</th>
<th>Herbivorous ducks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zooplankton</td>
<td>Macro-epiphytes</td>
<td>Suspension-feed polychaetes</td>
<td>Detritus-feed crustaceans</td>
<td>Spider crab</td>
<td>Blue crab</td>
<td>Predatory shrimp</td>
<td>Detritus-feed polychaetes</td>
<td>Epiphyte-grazing gastropods</td>
<td>Other gastropods</td>
<td>Tongue fish</td>
<td>Gulf flound &amp; needlefish</td>
<td>Atlantic silverside &amp; bay anchovies</td>
<td>Pinfish</td>
<td>Spot</td>
<td>Pipefish &amp; seahorses</td>
<td>Sheepshead minnow</td>
<td>Red Drum</td>
<td>Southern hake &amp; sea robins</td>
<td>Gobies &amp; blennies</td>
<td>Killifish</td>
<td>Herbivorous ducks</td>
</tr>
<tr>
<td>Zooplankton</td>
<td>Macro-epiphytes</td>
<td>Suspension-feed polychaetes</td>
<td>Detritus-feed crustaceans</td>
<td>Spider crab</td>
<td>Blue crab</td>
<td>Predatory shrimp</td>
<td>Detritus-feed polychaetes</td>
<td>Epiphyte-grazing gastropods</td>
<td>Other gastropods</td>
<td>Tongue fish</td>
<td>Gulf flound &amp; needlefish</td>
<td>Atlantic silverside &amp; bay anchovies</td>
<td>Pinfish</td>
<td>Spot</td>
<td>Pipefish &amp; seahorses</td>
<td>Sheepshead minnow</td>
<td>Red Drum</td>
<td>Southern hake &amp; sea robins</td>
<td>Gobies &amp; blennies</td>
<td>Killifish</td>
<td>Herbivorous ducks</td>
</tr>
</tbody>
</table>
```

#### Notes:
- The matrices show the carbon flow interactions between different trophic levels.
- Row normalized represents the rows as the source of carbon flow, while column normalized represents the columns as the source.
9.6.2 Obtaining partitions with original and modified versions of REGGE

Nordlund (2007: 2) argued that both general approaches to identifying regular blocks (treating all non-empty blocks as regular or using a global cutoff value) are problematic for valued networks with great value spans. While I strongly support his arguments, I believe they should not be limited to identifying regular blocks (distinguishing them from null), but also to finding the appropriate partition. If we look at Figure 9.43 and Figure 9.44 (if we ignore that the null blocks are indicated by the dashed cells), it is really hard to decide which block should be null and which regular. Therefore, I modified REGGE \(^\text{98}\) to look at the two normalized matrices, one when assessing the similarity of the incoming and one when assessing the similarity of the outgoing arcs. The version of REGE obtained is called REGGE-NM.

The approach used by Luczkovich et al. (2003) results in a partition with a large number of clusters, which is due to their criteria for obtaining the number of clusters. Although I tested their approach, the difference between the two REGGE algorithms is not as obvious at such a high resolution. In addition, in at least some applications fewer clusters are desired. To achieve this, I selected a different approach to select the number of clusters that is very commonly used with hierarchical clustering, namely looking for a relative large vertical \(^\text{99}\) separation on the dendrogram between two consecutive joints compared to the next separation. I also used another hierarchical clustering method, Ward's hierarchical clustering, as it is known to produce relatively similarly sized clusters. This algorithm was chosen to avoid obtaining clusters with only a few units (e.g., 1 or 2). The dendrograms obtained by Ward's hierarchical clustering on classical REGE similarities and those obtained with the REGGE-NM algorithm that uses normalized matrices (see Subsection 4.4.3. for a detailed description of the algorithm) are presented in Figure 9.45.

Although the largest ‘jump’ appears in two clusters, more have been selected. The last relatively high jump appears in both dendrograms at five clusters. This is even more clearly seen in Figure 9.46 where only the screeplot for the heights of joints is presented. Based on that, five clusters were selected.

The partitioned normalized matrices obtained using the partitions obtained by Ward's hierarchical clustering are presented in Figure 9.47 and Figure 9.48.

9.6.3 Interpretation and reduced graphs – original REGGE

The partitions can be directly interpreted based on the partitioned matrices presented before. Let us first look at the partition obtained by the original REGGE (Figure 9.47).

\(^{98}\) For more details, see Subsection 4.4.3. Another version of REGE, REGGE-OWNM was also tested. It produced almost identical (they differ in the cluster membership of only one unit) five-cluster partition as REGGE-NM.

\(^{99}\) Vertical as in Figure 9.45.
The first cluster consists of flora (plants), some species that feed only on flora (from this cluster) and all versions of organic carbon. The fauna in this cluster is an important consumer of the flora and the flora is an important food source of fauna in this cluster. They are the most important food source for all clusters but the last one. The Sediment POC is the most important terminal state for most species with the exception of the last cluster (the species of the last cluster are birds and apparently they do not (usually) die in this ecosystem).
Figure 9.47: Partitioned normalized matrices obtained using the partitions obtained by Ward's hierarchical clustering on 1 - original REGGE similarities.

Figure 9.48: Partitioned normalized matrices obtained using by Ward's hierarchical clustering on 1 - REGGE-NM similarities

Legend

Row normalized

Column normalized

Legend
Most members of the second cluster feed primarily on the first cluster, while some feed primarily on the third. Some also feed on members of their own cluster. On the other hand, members of this cluster are not very important as predators, with the exception of their own cluster. They are a relatively important food source for the fifth and partly the fourth clusters.

The third cluster feeds almost exclusively on the first cluster and is, on average, the second most important food source of the second, fourth and fifth clusters.

The fourth cluster feeds primarily on the first, but also on the third and second clusters. A member from the third cluster is an important predator of most members of this cluster. The fifth cluster consists entirely of birds, mainly predators. They are not important predators of any cluster, only Raptors are the only terminal state of one member of this same cluster. They mainly feed on the second and third clusters.

While all of this can be deducted from the partitioned normalized matrices, a more simplistic picture is desired. In blockmodeling, an image or reduced graph is usually used for this purpose. In the reduced graph, the units represent clusters and the ties represent (other than null) blocks, while null blocks are indicated by the absence of ties. To construct a reduced graph, the blocks must be classified into null and regular. Nordlund (2007) suggested one approach to obtaining blockmodels. Using the same method that he employed in his paper, the following percentages of fulfilled criteria for regular blocks were computed.

\[
\begin{array}{ccccc}
[1,] & [2,] & [3,] & [4,] & [5,] \\
[1,] & 0.82 & 0.64 & 0.67 & 0.38 & 0.06 \\
[2,] & 0.60 & 0.54 & 0.33 & 0.38 & 0.45 \\
[3,] & 0.62 & 0.42 & 0.60 & 0.50 & 0.25 \\
[4,] & 0.48 & 0.00 & 0.30 & 0.15 & 0.38 \\
[5,] & 0.00 & 0.00 & 0.00 & 0.00 & 0.17 \\
\end{array}
\]

Whenever a percentage is greater than or equal to 0.5 (a different threshold could also be selected), the block is classified as regular and as null otherwise. The first image graph presented on Figure 9.49 is drawn based on these values. In this graph, only those ties (percentages of fulfilled criteria) greater than or equal to 0.5 are drawn and the darkness and thickness of the tie corresponds to its value.

Nordlund (2007) noted that the fulfillment criteria for row and column regularity could be combined differently. The way he combined them involved more weight being given to the larger dimension. If we select one of the alternatives that he offers, that is to simply compute the mean of the row and column fulfillment criteria, the two dimensions are given equal weight regardless of their size. In this case, we obtain the following percentages of fulfilled criteria for regular blocks:
Consequently the image graph also changes, as it shown on the second image graph in Figure 9.49. I find the second image more appropriate since, based on the partitioned normalized matrices, I do not see why there should be ties from clusters 2 and 3 to cluster 1, yet not from cluster 4.

Figure 9.49: Image graphs obtained by Nordlund’s (2007) method on the partitions obtained by Ward’s hierarchical clustering on 1 – original REGGE similarities

In addition, some of approaches developed in this dissertation (Chapter 7) may also be used for that purpose. These approaches are binary, valued and implicit\textsuperscript{100} blockmodeling. Further, the averaging rules suggested in Section 7.4, in conjunction with a cut-off value, could also be used. It should be noted that binary, valued and implicit (generalized) blockmodeling use a different approach to the identification of null and regular blocks. They do not compute some statistics in rows and columns of the block in question and then use some threshold to decide if the block is regular or not. Instead, they compute the inconsistencies of the empirical blocks with both ideal null and regular blocks. Then the ideal blocks with smaller inconsistency are selected. In relatively sparse networks (or in networks with very few ties with large values) such as this one this way of determining a (regular) blockmodel (classification of regular and

\textsuperscript{100} There might be problems with implicit blockmodeling as it has been shown that it often does not perform well when null blocks are allowed, which is necessary for the classification.
null blocks) is more in favor of null blocks\textsuperscript{101} than those proposed by Nordlund (2007). This is especially true for larger blocks.

Here only valued blockmodeling and averaging rules are used. Binary blockmodeling is not used as it employs less information than valued blockmodeling, while implicit blockmodeling does not perform satisfactorily due to its problems when allowing null blocks in networks with tie values distribution with very long right tails. Valued blockmodeling uses $f$-regular equivalence, and the suitable $f$ functions are the maximum and sum. Also, valued blockmodeling requires we select another parameter; parameter $m$. Using different $f$ functions (maximum and sum) requires different values of parameter $m$.

When using maximum as $f$ function, the reasonable $m$ value is $2/51$, that is twice the cut-off value suggested by Nordlund (2007). The reason for using such a cut-off value is that in valued blockmodeling (with the maximum as the $f$ function), for each cell it is checked whether its row (or column) maximum is closer to $m$ or if its cell value is closer to 0. With this approach, we get the following image:

\[
\begin{bmatrix}
[1,]  "reg"  "reg"  "reg"  "null"  "null" \\
[2,]  "reg"  "null"  "null"  "null"  "reg" \\
[3,]  "reg"  "null"  "null"  "null"  "null" \\
[4,]  "reg"  "null"  "null"  "null"  "null" \\
[5,]  "null"  "null"  "null"  "null"  "null"
\end{bmatrix}
\]

In fact, we can compute for each cluster at which value of parameter $m$ it would become max-regular (and not null). This is done for a sequence of values from 0.001 to 1 with steps of 0.001. The matrix obtained in this way is as follows:

\[
\begin{bmatrix}
[1,]  0.288  0.058  0.114  0.038  0.006 \\
[2,]  0.065  0.031  0.009  0.020  0.046 \\
[3,]  0.085  0.025  0.035  0.034  0.023 \\
[4,]  0.067  0.000  0.018  0.001  0.015 \\
[5,]  0.000  0.000  0.000  0.000  0.017
\end{bmatrix}
\]

If a block has a value lower than $2/51 \approx 0.039$, it is classified as null in the previous matrix and as max-regular otherwise.

The first reduced graph in Figure 9.50 is obtained using this matrix where values lower than $2/51$ (the null blocks) are replaced by 0s.

Although using the maximum as the $f$ function value is more compatible with REGGE, the sum was also considered. This corresponds to the idea that it is irrelevant what the maximum proportion is that any member of a cluster represents in a predator’s diet, only the proportion

\textsuperscript{101} If we believe that this is inappropriate, it is possible to penalize the null blocks by multiplying their inconsistency by a certain (larger than 1) value and thus making them less favorable.
that all members of that cluster represent in its diet. Of course, something similar could be said for the proportion of a ‘pray’ eaten by members of a certain cluster. When using the sum as the \( f \) function, one possible value for parameter \( m \) would be \( 1/5 \) (the number of clusters) or even twice that value. However, such a large \( m \) value results in only one sum-regular block. The \( m \) value should be lowered to almost the one used where maximums are the function \( f \). This seems to indicate there is usually only one relatively high value in a given row and block. Due to the similarity of the results with those obtained using the maximum as the function \( f \) value, the results of using the sum are not presented.

**Figure 9.50**: Image graphs obtained using valued blockmodeling and averaging rules on the partitions obtained by Ward's hierarchical clustering on 1 – original REGGE similarities

Based on these \( m \) values calculated both according to \( max \)-regular and \( sum \)-regular equivalence and from the appearance of the blocks in Figure 9.47, we can also speculate that of most blocks that are considered \( f \)-regular, blocks are probably not considered as \( f \)-regular due to their small inconsistency with the ideal \( f \)-regular block but rather due to their (sometimes extremely high) inconsistency with the ideal null block.

Next the averaging rules suggested as being more general in Section 7.4.6 are used. Two options are considered. One is the mean of the mean row and mean column maximums and the other is the mean of the mean row sums and mean column sums. The following matrix was obtained using the mean of the mean row maximums and mean column maximums:

\[
egin{array}{cccccc}
1 & 2 & 3 & 4 & 5 \\
1 & 0.52 & 0.23 & 0.26 & 0.22 & 0.06 \\
2 & 0.33 & 0.13 & 0.05 & 0.09 & 0.20 \\
3 & 0.36 & 0.08 & 0.11 & 0.07 & 0.12 \\
4 & 0.37 & 0.00 & 0.12 & 0.01 & 0.03 \\
5 & 0.00 & 0.00 & 0.00 & 0.00 & 0.09
\end{array}
\]
The reduced graph obtained using this matrix with values lower than 0.2 replaced by 0s is the second graph in Figure 9.50. The results for using the mean of the row and column sums are very similar and are therefore not presented.

In Figure 9.49 and Figure 9.50, we have four different reduced graphs and the question arises of which one is the most appropriate. I will not give a straightforward answer to that question. However, I can say that the reduced graphs in Figure 9.49 are only based on the pattern of the ties/cells in the normalized networks/matrices with values over 1/51, while the one in Figure 9.50 also considers the values of ties, but in two different ways. The first one determines a blockmodel based on whether the cells in the blocks (of the normalized matrices) are closer to 0 or the row and column maximums are ‘closer’ (only if they are lower) to 2/51. Since the reduced graphs are based on different approaches, the tie values also require a different interpretation. The tie values in the reduced graphs in Figure 9.49 represent how close the tie pattern is to the ideal pattern for regular ties, while in Figure 9.50 they are more connected to the actual flow of carbon between and within clusters. In the first graph in Figure 9.50 they represent the parameter $m$ values where the classification of a certain blocks changes from null to regular, while in the second graph the interpretation is more straightforward – the values are simply the means of the mean row and mean column maximums.

As the interpretation of all four reduced graphs (while acknowledging the differences described in the previous paragraph) can be done in very similar way, only one interpretation is presented, namely the interpretation of the last (second) graph in Figure 9.50. The strongest carbon flow (in terms of the row and column maximums in the normalized networks) occurs within cluster 1. Also, there are strong carbon flows from all other clusters except cluster 5 to cluster 1. However it should be noted that this due to only one unit in cluster 1, that is to Sediment POC, a unit that does not really fit into any cluster as it is not a living being in contrast to most other units. More meaningful are the carbon flows from cluster 1 to all clusters except cluster 5, as they constitute a feeding relationship. Also, the units of cluster 5 feed on the units of cluster 2.

### 9.6.4 Interpretation and reduced graphs – modified REGGE

The partition obtained with the modified REGGE (REGGE-NM) gives a cleaner partition of ties into blocks and therefore also has a simpler interpretation. Cluster 1 now consists entirely of flora (or at least not classical animals). Therefore, they do not feed on any cluster whereas clusters 2, 3 and partly 4 feed on it.

Clusters 2 and 3 consist almost exclusively of herbivores and feed only on cluster 1. Cluster 3 also feeds in a very small quantity on itself. The unit Sediment POC located in cluster 2 is again an outlier, as is the end state of most units in all clusters except the last one. A relatively large portion of cluster 2 is eaten by cluster 4, while the rest (still the majority) usually end up as Sediment POC (a member of cluster 2). The situation is similar for cluster 3, save that cluster 4 is not such an important predator.
Cluster 4 consists mainly of aquatic predators and they feed primarily on cluster 3, and in part on clusters 2 and 4 (itself). The majority of cluster 4 ends up as Sediment POC (cluster 2), although some members are also eaten by its own members and those of cluster 5.

Cluster 5 consists of predatory birds. They primarily feed on cluster 4, but also in smaller quantities on clusters 2 and 3. Their ‘end state’ is not seen in this ecosystem.

In the previous subsection it was demonstrated how these different kinds of reduced graphs are obtained. Therefore, this step is not repeated here. The reduced graphs themselves are presented in Figure 9.51 and Figure 9.52.

*Figure 9.51: Image graphs obtained by Nordlund’s (2007) method on the partitions obtained by Ward's hierarchical clustering on 1 – REGGE-NM (modified REGGE) similarities*

![Nordlund](image)

*Figure 9.52: Image graphs obtained using valued blockmodeling and averaging rules on the partitions obtained by Ward's hierarchical clustering on 1 – REGGE-NM (modified REGGE) similarities*

![Valued blockmodeling](image)

![Averaging rules - values over 0.2 mean of row/column maximums](image)
Again, let us conclude this subsection with an interpretation of the last reduced graph (the second in Figure 9.52). Carbon flows from cluster 1 to clusters 3 and 4, while some carbon also returns to cluster 1 from cluster 2 (exclusively due to the bacteria feeding on DOC and Sediment POC). Then carbon flows from clusters 2 and 3 to cluster 4. There are also carbon flows within cluster 2 and from clusters 3 and 4 to cluster 2, however this is entirely due to Sediment POC (from cluster 2) being the most important end-state of most units in all clusters except cluster 5. The last remaining carbon flow is from cluster 4 to cluster 5.

9.6.5 Conclusion

In this section an alternative analysis to that presented in Luczkovich et al. (2003) and Nordlund (2007) of the St. Marks carbon flow network (Baird et al., 1998) is presented. The aim of the analysis in this example is to obtain a partition of the St. Marks food web into a smaller number of clusters (the original analysis presented a 10-cluster solution) and to apply a modified version of REGGE developed in Section 4.4.3 (REGGE-NM). The REGGE-NM (modified version of REGGE) does not compare the absolute values of ties, but rather the relative values tie compared to the sum of values of all incoming/outgoing ties of a given unit. I believe this is more appropriate in circumstances like this one where the tie values are very much influenced by the ‘size’ (in some sense – in the case of the St. Marks food web ‘size’ corresponds to the total amount of carbon contained in a certain unit) of the units. I believe that such patterns are more relevant since they carry more information about the structure of the food web. When comparing the results based on this point of view, REGGE-NM produces better results, namely clearer blocks induced by the partition and more interpretable partitions. Unfortunately, this approach also does (at least for the selected number of clusters) not separate several versions of organic carbon from living beings.

In addition, Nordlund's (2007) suggestions, valued blockmodeling and averaging rules (with a cut-off value) were also applied for the purpose of obtaining regular blockmodels and for obtaining image graphs.

Another possibility that was not explored is to make two blockmodels, one using null and row-regular block types in row-normalized matrix/network and one using null and column-regular block types in column-normalized matrix/network. This would produce two blockmodels/image graphs, one representing the important receivers of carbon from a given cluster from that cluster's point of view and the important sources of carbon for a given cluster from that cluster's point of view. However, this exceeds the objectives of this example which were to obtain a partition into fewer clusters and to assess the modified version of REGGE.

9.7 Conclusion and lessons learned

In the previous six sections, the use of several blockmodeling techniques is demonstrated in real and artificial valued networks. The emphasis is usually on the generalized blockmodeling approaches, especially those for the blockmodeling of valued networks. The exceptions are Sections 9.2 and 9.6 where different versions of the REGE algorithm are examined.
The first two sections (Sections 9.1 and 9.2) are devoted to demonstrating how the generalized blockmodeling approaches (Section 9.1) and the REGE algorithm (Section 9.2) work in very simple artificial valued networks. The remaining examples are empirical valued networks that were selected to demonstrate and evaluate the performance of the reviewed approaches. This conclusion is mainly based on these last four examples, since those in the first two sections were only really presented for demonstration purposes.

Several properties or characteristics of the reviewed approaches or their implementations have been learned or confirmed in this chapter. By way of a quick overview, we can confirm that the direct approaches usually provide better partitions than the indirect ones, with the added value of also providing a value of the criterion function. The main problem that any researcher attempting to use these approaches in real networks will face is which approach, especially out of the direct, generalized blockmodeling approaches, is the most suitable. Unfortunately, little guidance can be given on this topic. Ideally, the researchers would select the approach whose definition of some equivalence or ideal blocks best matches the theoretical concept that they want to use to obtain partitions. However, this is rarely possible in a more exploratory analysis.

Currently, binary blockmodeling has a unique advantage in terms of implementation as it is the only approach implemented in a relatively fast program (in Pajek 1.11 (Batagelj and Mrvar, 2006)). As such, it is the only generalized blockmodeling approach that can be effectively (in reasonable time) used for networks with more than 10 to 20 units. Other generalized approaches can be used in networks with up to about 50 units if we are prepared to wait for several hours (or even a day) for the results. The results presented in this chapter also indicate that it usually also provides reasonable partitions if the slicing threshold is selected appropriately. There are some exceptions. For example, binary blockmodeling according to the regular equivalence of a network of institutions based on their cooperation with social workers (Section 9.3) does not produce a reasonable partition. One problem that occurs more often is that binary blockmodeling, especially according to regular equivalence, produces a larger number of optimal partitions (these partitions have the same value of the criterion function, which is also the lowest value out of all those computed). Instances of this can be seen in several examples in this chapter (Sections 9.1, 9.3 and to a smaller extent in Section 9.4). This is due to the fact that binary blockmodeling measures the inconsistencies of empirical blocks with ideal blocks less precisely since it only uses binary (instead of interval valued) data.

Valued blockmodeling also suffers from this problem (see Sections 9.1 and 9.3), albeit to a smaller extent. Both of these approaches also require that a parameter (slicing threshold \( t \) or parameter \( m \)) is selected in advance. Where we do not have a substantive reason for selecting a certain value this could prove problematic.

Homogeneity blockmodeling does not suffer from either the problem of the imprecise measurement of inconsistencies or of having a parameter (besides the measure of variability and number of clusters) that must be selected in advance. However, it suffers from the
so-called ‘null block problem’ (see Section 7.4). The core of this problem is that the ideal null block as defined in homogeneity blockmodeling is also a special case of almost all other block types. This brings two undesirable effects that may be observed in the examples given in this chapter. The first is that homogeneity blockmodeling by itself (other procedures can be used for that purpose) cannot classify blocks as null as opposed to some other block type. The effect is also that homogeneity blockmodeling (especially sum of squares blockmodeling) often produces less clear ‘null blocks’ \(^{102}\). This is especially noticed in examples where the network of institutions based on their cooperation with social workers (Section 9.3) and the network of borrowing notes (Section 9.4) were analyzed. Some ideas that might reduce or eliminate this problem are suggested in Section 7.4. In homogeneity blockmodeling the compatibility of some block types is questionable, although this problem is not addressed in this chapter.

Implicit blockmodeling does not suffer (in its original version, meaning with null blocks) from any of the problems described above. However, when applied to the empirical networks in this chapter its performance was usually not satisfactory. In these examples, the partitions obtained using this approach were often reasonable, the most notable exception being the partition obtained when applied to the network of the ‘sweet’ island (Section 9.5). In other examples, even when the partition is reasonable, the obtained blockmodel is not. Many blocks that should be classified as complete or regular are classified as null. This is caused by the fact that block inconsistencies (in implicit blockmodeling) are actually relative to the block maximum \(^{103}\). One way to improve the partitions obtained is to not use the null blocks as allowed blocks and to make them, as in homogeneity blockmodeling, special cases of most other block types. We could say this transforms implicit blockmodeling into homogeneity blockmodeling with a strange measure of variability. This does improve the partitions obtained using this approach, but it also means that the null block problem applies. The effect that the null blocks are less clear is not as serious.

Broadly speaking, most generalized blockmodeling usually performed well, although binary blockmodeling sometimes provided less suitable partitions or a larger number of partitions. One of the valued blockmodeling partitions was often selected as the most suitable one, but this might also be partly due to the fact that several values of parameter \(m\) were used and the best partitions were then selected. Implicit blockmodeling produces good partitions if used without null blocks.

The indirect approaches perform relatively well in most examples, although usually not as well as the generalized blockmodeling approaches. In some cases, the indirect approaches are able to identify reasonable partitions at some, but not at all levels of clusters. Such situations occur when the indirect approach according to structural equivalence was used on the valued

---

\(^{102}\) As mentioned above, homogeneity blockmodeling does not classify blocks as null. The classification is usually done on block averages or some similar statistics based on tie values in a block.

\(^{103}\) In a few block types, also to the maximum of a certain row or column.
network of organizations based on their cooperation with social workers (Section 9.3) and on the network of borrowing notes (Section 9.4). Indirect approaches according to regular equivalence face similar problems when applied to the network of the ‘sweet’ island (Section 9.5). This last one is a particularly interesting case. Usually, indirect approaches find good partitions into fewer clusters (2) and have problems with more detailed partitions, where often only the ‘core’ cluster is well identified. However, in this case the partition into three clusters is reasonable while the one into two is not. Due to the speed of their execution, their use is strongly supported at least as a preliminary step (to assess the appropriateness of the structural or regular partitions and of the number of clusters).

As mentioned, generalized blockmodeling usually performed better. However, there is one type of network where the generalized blockmodeling approaches do not perform well, while a version of REGE, especially designed for such networks, provides good partitions. This type of network is characterized by the tie values that are in some sense dependent on the ‘size’ of the units. An example of such a network is the network of St. Marks carbon flow (Section 9.6). Generalized blockmodeling might provide reasonable partitions if applied to a suitably normalized network but they were unable to provide such a partition when used simultaneously on row and column normalized matrices.

In two examples (the network of organizations based on their cooperation with social workers in Section 9.3 and the network of St. Marks carbon flow in Section 9.6) image graphs are constructed based on the partitions obtained. Several rules are used for obtaining these image graphs. The averaging rules based on sum of squares blockmodeling and valued blockmodeling probably produce the best image graphs. Those based on the sum of squares approach give more emphasis to tie values in the block (while taking into account the type of block), while those based on valued blockmodeling give more emphasis to the pattern of ties (while taking into account how far the tie values are from 0 and from parameter $m$).
10 EVALUATION OF DIRECT AND INDIRECT BLOCKMODELING OF REGULAR EQUIVALENCE BY SIMULATIONS

The aim of this chapter is to compare different approaches to generalized blockmodeling and some versions of REGE on artificially generated valued networks that should be approximately max-regularly equivalent. The generated networks were based on different known (max-)regular blockmodels and partitions. The obtained partitions were compared to the original (known) partition using the Adjusted Rand Index.

A smaller part of the chapter is dedicated to analyzing networks generated according to blockmodels that also include other block types, more precisely row- and column-dominant block types, in addition to the (max-)regular and null.

10.1 Design of the simulations

All simulations were done by repeating the following general procedure for a given number of times for each setting:

- Generation of a valued network based on a blockmodel and other parameters
- Application of different methods
- Comparing the obtained partition(s) with the original partition using the Adjusted Rand Index (Hubert and Arabie, 1985: 198)

The simulations were accomplished in two stages. The different settings used in Stage 1 are presented in Subsection 10.2.1 while those used in Stage 2 are found in Subsection 10.3.1. They include the blockmodel and parameters used to generate the tie values.

As we noted above valued networks were generated. The procedure used for this purpose is explained in Subsection 10.1.2. Then each valued network was submitted to several different methods, which are presented in Section 10.1.1.

10.1.1 Evaluated methods

The blockmodeling methods described below were evaluated by the simulations (although some were not evaluated by each simulation). The text in bold are the labels used to identify a certain method. Versions of the methods are described by the series of labels described below, separated by the symbol ‘|’. The labels appear in the same order as they are introduced below.
Indirect methods:

- **sedist** – Ward’s hierarchical clustering on distances computed using the Corrected Euclidean-like dissimilarity (Burt and Minor, 1983 in Batagelj et al., 1992: 71) with $p = 2$.

$$d_s(X_i, X_j) = \sqrt{\sum_{k=1}^{n} \left( (r_{ik} - r_{jk})^2 + (r_{ik} - r_{kj})^2 \right) + 2 \left( (r_{ii} - r_{jj})^2 + (r_{ij} - r_{jj})^2 \right)}$$

- **str** – structural equivalence (not an option/parameter as this approach can only be used for structural equivalence)

- **REGGE, REGGE-OW, REGDI, REGDI-OW** - Several versions of REGE (see Section 4.4 for the meaning of the following acronyms): REGGE, REGGE-OW, REGDI, REGDI-OW. In all versions, the number of iterations was set to 3.

- **reg** – regular equivalence (not an option/parameter as REGE can only search for regular equivalence)

Direct methods (generalized blockmodeling):

- **bin** - Binary blockmodeling using options:
  - Equivalence (allowed blocks):
    - **reg** – regular equivalence (null and regular)
    - **str** – structural equivalence (null and complete)
    - **pre** – pre-specified blockmodeling: the blockmodel used in the generation of the binary network was used as the pre-specified blockmodel instead of only specifying the allowed blocks.
  - Values of the threshold $t$ used for binarization:
    - **halfmax** - half of the empirical maximum of tie values
    - **min** - the second (the first one is usually 0) minimum of the empirical distribution of all tie values as indicated by Figure 10.1.

- **val** - Valued blockmodeling using options:
  - Equivalence (allowed blocks):
    - **reg** - $f$-regular equivalence at level $m$ (null and $f$-regular)
    - **pre** – pre-specified blockmodeling: the blockmodel used in generation of the binary network was used as the pre-specified blockmodel instead of only specifying the allowed blocks.
- **max** – Function \( f \) used in \( f \)-regular blocks (not used with structural equivalence)

- Values of the parameter \( m \) used
  - **max** - the empirical maximum of tie values
  - **2min** - twice the second (the first one is usually 0) minimum of the empirical distribution of all tie values as indicated by Figure 10.1.

- **cen** - Censoring the tie values at the value of parameter \( m \) (either used or not). If it appears next to the method then it was used.

*Figure 10.1: The second density minimum*

- Homogeneity blockmodeling using options (the labels depend on the type of homogeneity blockmodeling and are presented below):
  - Type of homogeneity blockmodeling (or measure of variability used):
    - **ss** – Sum of squares blockmodeling (sum of squares deviations of values from their mean)
    - **ad** – Absolute deviations blockmodeling (sum of absolute deviations of values from their median)
  - Equivalence (allowed blocks):
    - **reg** - \( f \)-regular equivalence (null and \( f \)-regular)
    - **str** – structural equivalence (null and complete)
  - **max** or **mean** – Function \( f \) used in \( f \)-regular blocks (not used with structural equivalence)
• **imp** - Implicit blockmodeling using options:
  - Equivalence (allowed blocks):
    - **reg** - $f$-regular equivalence (null and $f$-regular)
    - **pre** – pre-specified blockmodeling: the blockmodel used in generation of the binary network was used as the pre-specified blockmodel instead of only specifying the allowed blocks.
  - **max** – Function $f$ used in $f$-regular blocks (not used with structural equivalence). Not really an option since only the maximum can be used with implicit blockmodeling, however it is shown in results as a reminder that this function is actually used.

• Options for all direct approaches:
  - Search procedures:
    - Local optimization with 20 random starting partitions (this and the next option are the default ones and are therefore not marked. Which one is used is evident from the accompanying text, usually defined for the whole stage)
    - Full search – checking all possible partitions (see the comment on the previous option)
    - **100** – Optimization of 100 random starting partitions
    - **OC** – Optimization of the *correct* (the one used in generation of the network) partition as the starting partition
    - **Cor** – No optimization – the label indicates the results (the inconsistency\(^{104}\), as the Adjusted Rand Index is 1 by default) for the *correct* partition

As mentioned, versions of the blockmodeling methods are described by the series of labels that are described above. The labels appear in the same order as they are introduced. For example, a valued blockmodeling according to *max*-regular equivalence where $m$ is selected as the empirical maximum of the tie values without censoring is referred to as ‘val|reg|max|max’. If censoring had also been used, the version's label would be ‘val|reg|max|max|cen’.

All the evaluated methods are implemented in the `blockmodeling 0.1.2` package (Žiberna, 2006) and this implementation was used for all methods. The limitations that the use

\(^{104}\) The inconsistency stands for the total inconsistency of the partition with an equivalence (can be generalized), which is also the value of the criterion function (in generalized blockmodeling)
of this still experimental software presents for the results obtained in this chapter are
described in Section 10.5.

10.1.2 Generation of networks
The networks were generated based on several parameters:

- partition
- blockmodel
- parameter controlling the enforcement of strict regularity in the binary network (enforce)
- parameters of the beta distribution used for generation of the tie values; one of them
  was always the same for the whole network, while the other could be block-specific
  (in terms of the position of the block)
- multiplication factor (can be block-specific)

The more detailed procedure for generating a network is as follows:

1. Generation of a binary network based on a partition, a blockmodel, and the parameter
   controlling the enforcement of strict regularity.
   a. the partition was used to split a (empty) network (a matrix of 0s) into blocks
   and to determine the size of the network
   b. with blocks where the blockmodel indicated null blocks, nothing was changed
   c. in regular blocks, each cell had a probability of becoming 1 equal to p:

   \[ p = \frac{1}{\min(n_r, n_c) - 1}, \]

   where:
   - \( n_r \) is the number of rows in the blocks
   - \( n_c \) is the number of columns in the blocks
   d. if regularity was enforced the block was checked for regularity, that is, each
      row and each column were checked if they had at least one 1 (tie). If not, 1 was
      added to a randomly chosen cell from that row/column

2. Generation of a valued network based on the binary network and the remaining
   parameters (beta parameters and multiplication factor).
   a. Based on the binary network, the valued network was generated from the beta
      distribution. Beta distribution has the density:

      \[ f(x) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)}x^{a-1}(1-x)^{b-1} \]

      where \( a \) and \( b \) are two shape parameters and \( \Gamma \) is the Gamma function. It can
      have positive values on the interval \([0, 1]\).

      The values of the parameters \( a \) and \( b \) depend on the type of tie in the binary
      network (0 or 1) and two additional parameters, \textit{shape1} and \textit{shape2}. The effect
      of the parameters of the beta distribution \((a \text{ and } b)\) or the \textit{shape1} and \textit{shape2}
      parameters on the shape of beta distribution can be seen in Figure 10.2 and
      Figure 10.3.
For ‘0’ ties, the values were generated from the beta distribution with parameter $a$ always set to 1 and the parameter $b$ set to $shape1$. For ‘1’ ties, the values were generated from the beta distribution with parameter $a$ set to $shape1$ and the parameter $b$ set to $shape2$, which could be block specific. The parameter $shape2$ was set to 1 in the most basic version, making the distribution for ‘1’ ties mirror image of the distribution for ‘0’ ties. The other value used for the parameter $shape2$ was 4. The parameter $shape1$ could take on values 10, 8, 6, 4 and 2, but was often restricted to valued 8 and 4. The values of the parameters $shape1$ and $shape2$ in individual settings are specified in Section 10.2.

Figure 10.2: The beta distributions used to generate the ‘0’ ties

![Beta distributions for '0' ties](image1)

Figure 10.3: The beta distributions used to generate the ‘1’ ties

![Beta distributions for '1' ties](image2)
b. All values in certain blocks were then multiplied by the multiplication factor \((mf)\). Often, this was the same for all blocks and therefore made no impact; however it could be block-specific.

The way the networks were generated has a large affect on the results. The limitations of the results due to the way the networks were generated are discussed in Section 10.5.

### 10.1.3 Comparing partitions

The result of each blockmodeling method is a partition (or sometimes a set of partitions). These were then compared with the original partition used in the generation of the networks using the Adjusted Rand Index (Hubert and Arabie, 1985: 198). The Adjusted Rand Index is the Rand Index (Rand, 1971 in Hubert and Arabie, 1985: 195), adjusted for chance. The Rand Index is computed as the number of pairs of units that are in the same cluster in both partitions, divided by the number of all possible pairs of units. Based on this, the Adjusted Rand Index is computed as (Hubert and Arabie, 1985: 198):

\[
\text{Adjusted Rand Index} = \frac{\sum_{i=1}^{g} \sum_{j=1}^{C} \binom{n_{ij}}{2} - \sum_{i=1}^{g} \left( \sum_{j=1}^{C} \binom{n_{ij}}{2} \right) \left( \frac{n}{2} \right) \left( \frac{n}{2} \right)}{\frac{1}{2} \left( \sum_{i=1}^{g} \binom{n_{i}}{2} + \sum_{j=1}^{C} \binom{n_{j}}{2} \right) - \sum_{i=1}^{g} \sum_{j=1}^{C} \binom{n_{ij}}{2} \left( \frac{n}{2} \right) \left( \frac{n}{2} \right)}
\]

As mentioned, the Adjusted Rand Index computed by comparing the partition obtained using blockmodeling methods and the original partition is used as a measure of the ability of the blockmodeling method to find the correct partition and therefore of its quality. As a single measurement would be unreliable and could be heavily influenced by random factors built into the procedure used for generating the networks, for each different setting \(ngen\) networks were simulated and \(ngen\) measurements (one for each network) of the Adjusted Rand Index for each method (or more precisely each version of each method) were obtained. Actually, generalized blockmodeling can produce several partitions as a result of the optimizational process. In such cases, the mean of the Adjusted Rand Indices for the partitions obtained on each network was first computed\(^{105}\). Then, the mean of these Adjusted Rand Indices over all networks was computed for each method.

---

\(^{105}\) In case when there were more than 50 ‘optimal’ partitions, only the first 50 were taken into account. In addition, one additional partition was considered for each starting point.
Table 10.1: Notation used to compute the Adjusted Rand Index

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Partition $V = {V_1, V_2, ..., V_C}$</th>
<th>Sums</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td>$n_{11}$, $n_{12}$, ..., $n_{1C}$</td>
<td>$n_{1}$</td>
</tr>
<tr>
<td>$U_2$</td>
<td>$n_{21}$, $n_{22}$, $n_{2C}$</td>
<td>$n_{2}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$U_R$</td>
<td>$n_{R1}$, $n_{R2}$, ..., $n_{RC}$</td>
<td>$n_{R}$</td>
</tr>
<tr>
<td>Sums</td>
<td>$n_{.-}$, $n_{.-}$, ..., $n_{.-}$</td>
<td>$n_{..} = n$</td>
</tr>
</tbody>
</table>

Legend:
- $n_{ij}$ ... the number of units that belong to cluster $U_i$ in partition $U$ and to cluster $V_j$ in partition $V$.
- $n_{i.}$ ... the number of units that belong to cluster $U_i$ in partition $U$, which is also equal to $\sum_{k=1}^{C} n_{ik}$.
- $n_{.j}$ ... the number of units that belong to cluster $V_j$ in partition $V$, which is also equal to $\sum_{i=1}^{R} n_{ij}$.
- $n_{..} = n$ ... the number of all units.

10.2 Stage 1

The simulations were undertaken in two stages. In Stage 1, blockmodeling methods were evaluated on networks with only 11 vertices and in most cases with partitions into just 2 clusters, although in some cases partitions into 3 clusters were also used. This can be thought of as a preliminary stage where several ideas were tested. A full search was used in all settings with two-cluster partitions, while the results for a three-cluster partition were obtained through the optimization of 20 random partitions.

10.2.1 Settings

The basic characteristics of Stage 1 are:
- The number of units is 11.
- Two different partitions were used:
  - Two-cluster partition: 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2 and
  - Three-cluster partition: 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3

The two-cluster partition was used in many more cases.
- $shapeI$ parameter has values 10, 8, 6, 4, and 2.
- The parameter controlling the enforcement of strict regularity in the binary network ($enforce$) was set to TRUE and FALSE.
- The number of repetitions or generated networks ($ngen$) was set to 20. Based on each setting (combinations of parameters controlling the generation of networks), a network was generated 20 times.
In the case of two-cluster partitions, all possible partitions were checked when using generalized blockmodeling (a full or exhaustive search). In addition, a local search with 10 random starting partitions was used in some settings to compare the results. In the case of three-cluster partitions, only a local search with 20 random starting partitions was used.

The number of iterations in REGE was set to 3.

In all settings, all the methods described in Subsection 10.1.1 that do not use block size or maximum normalization were used. These are: ss|str, sedist|str, REGD|reg, REGD.ow|reg, REGE|reg, REGE.ow|reg, bin|reg|halfmax, bin|reg|min, bin|pre|halfmax, bin|pre|min, val|reg|max|2min, val|reg|max, val|pre|max|2min, val|pre|max|max, ad|reg|max, ss|reg|max, ad|reg|mean, ss|reg|mean, imp|reg|max, imp|wnull|reg|max, imp|pre|max

In some settings, the block size normalization was tested on all generalized blockmodeling methods and maximum normalization on implicit blockmodeling. The selected settings are indicated in Table 10.2 by ‘Norm’.

Other parameters (or those not ambiguously defined in the above list) used in generation of the networks are shown in Table 10.2. Values used to describe the blockmodel and parameters shape2 and mf in Table 10.2 (when not simple numbers) are explained in Table 10.3. When the value in Table 10.2 for parameters shape2 and mf is a simple number, it is the value of the parameter in all blocks. The most important distinction among the blockmodels used is that only in blockmodels 1T and 13 is the partition that is searched for maximal regular. In all others, the partition that is searched for is not maximal regular and this makes it much harder to find. However, the approaches suited for valued networks (all but binary blockmodeling) are usually quite successful at finding it if additional information is provided in the form of either different shape2 or mf parameters (which essentially makes the partitions that are searched for maximal regular in the valued sense).

Based on this, the settings are separated into four settings classes, named Clear pattern, Not maximal regular, Different dist. par. (Different distribution parameters), and Different block max. (Different block maximums). In the first class (Clear pattern) there are settings with either blockmodel 1T or 13. Here the partition that is searched for is maximal regular, even before the tie values are added (in the process of generating the network). The second setting, Not maximal regular, is quite the opposite. Here, the partition that is searched for is not maximal regular, even after the values are added. This makes this partition the hardest to find, especially for the REGE algorithms. In the last two settings, additional information is added to the binary networks which are generated in the same way as generated in class Not maximal regular. In the resulting networks, the partition that is searched for is maximal regular when the tie values are taken into account in most cases (the exception are settings "2|AR|1|D" and "2|AR|10|D"). This should make these two classes especially problematic for binary blockmodeling. In the case of the class Different dist. par., the additional information is provided in the form of the block-specific shape2 parameters, while in the case of the class Different block max., the additional information is provided in the form
of the block-specific multiplication factors \((mf)\). It is important to note that the information in the form of different multiplication factors is much stronger than that in the form of different \(shape2\) parameters. It also affects all block types, whereas the \(shape2\) parameters do not affect the null block type.

**Table 10.2: Settings used in the simulations – Stage 1**

<table>
<thead>
<tr>
<th>Id</th>
<th>(k)</th>
<th>blockmodel*</th>
<th>(shape2)*</th>
<th>(mf)*</th>
<th>settings class</th>
<th>additional methods</th>
<th>regularity enforced</th>
<th>regularity not enforced</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>CP</td>
<td>1</td>
<td>10</td>
<td>Not maximal regular</td>
<td>Norm</td>
<td>Norm</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>CP</td>
<td>4</td>
<td>10</td>
<td>Not maximal regular</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1T</td>
<td>1</td>
<td>10</td>
<td>Clear pattern</td>
<td>Norm</td>
<td>Norm</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1T</td>
<td>4</td>
<td>10</td>
<td>Clear pattern</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>C2P</td>
<td>G3</td>
<td>10</td>
<td>Different dist. par.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>C2P</td>
<td>4</td>
<td>G3</td>
<td>Not maximal regular</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>2T</td>
<td>G3</td>
<td>10</td>
<td>Different dist. par.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>BG</td>
<td>4</td>
<td>10</td>
<td>Not maximal regular</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>BG</td>
<td>1</td>
<td>10</td>
<td>Not maximal regular</td>
<td>Norm</td>
<td>Norm</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>CP</td>
<td>D</td>
<td>10</td>
<td>Different dist. par.</td>
<td>Norm</td>
<td>Norm</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>CP</td>
<td>R</td>
<td>10</td>
<td>Different dist. par.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>BG</td>
<td>R</td>
<td>10</td>
<td>Different dist. par.</td>
<td>Norm</td>
<td>Norm</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>CP</td>
<td>1</td>
<td>D</td>
<td>Different block max.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>CP</td>
<td>1</td>
<td>R</td>
<td>Different block max.</td>
<td>Norm</td>
<td>Norm</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>BG</td>
<td>1</td>
<td>R</td>
<td>Different block max.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>AR</td>
<td>1</td>
<td>D</td>
<td>Different block max.</td>
<td>Norm</td>
<td>Norm</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>AR</td>
<td>1</td>
<td>R</td>
<td>Different block max.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>AR</td>
<td>D</td>
<td>10</td>
<td>Different dist. par.</td>
<td>Norm</td>
<td>Norm</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>AR</td>
<td>R</td>
<td>10</td>
<td>Different dist. par.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>G3</td>
<td>G2</td>
<td>10</td>
<td>Different dist. par.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>3</td>
<td>2V1</td>
<td>G1</td>
<td>10</td>
<td>Different dist. par.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>3</td>
<td>13</td>
<td>1</td>
<td>10</td>
<td>Clear pattern</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>3</td>
<td>C</td>
<td>O4</td>
<td>10</td>
<td>Different dist. par.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>3</td>
<td>C</td>
<td>1</td>
<td>10</td>
<td>Not maximal regular</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:

\(k\) … the number of clusters

\(mf\) … multiplication factor

* the meaning of the values (in case of \(shape2\) and \(mf\) only those that are not numbers) is explained in Table 10.3.
### Table 10.3: Values (codes) in Table 10.2 (for Stage 1)

<table>
<thead>
<tr>
<th>blockmodel</th>
<th>C2P (Core and 2 Peripheries):</th>
<th>G3 (Group 3 different - disconnected):</th>
</tr>
</thead>
<tbody>
<tr>
<td>BG (Between Groups):</td>
<td>[1,] &quot;null&quot; &quot;reg&quot; [2,] &quot;reg&quot; &quot;null&quot;</td>
<td>[1,] &quot;null&quot; &quot;reg&quot; &quot;null&quot;</td>
</tr>
<tr>
<td>CP (Core-Periphery):</td>
<td>[1,] &quot;null&quot; &quot;reg&quot; [2,] &quot;reg&quot; &quot;null&quot;</td>
<td>[1,] &quot;null&quot; &quot;reg&quot; &quot;null&quot;</td>
</tr>
<tr>
<td>1T (1 Tie):</td>
<td>[1,] &quot;null&quot; &quot;null&quot;</td>
<td>[1,] &quot;null&quot; &quot;null&quot;</td>
</tr>
<tr>
<td>AR (All Regular):</td>
<td>[1,] &quot;null&quot; &quot;null&quot;</td>
<td>[1,] &quot;null&quot; &quot;null&quot;</td>
</tr>
<tr>
<td>R (Row):</td>
<td>4 4</td>
<td>1 1</td>
</tr>
<tr>
<td>2T (2 Ties):</td>
<td>1 1</td>
<td>1 1</td>
</tr>
<tr>
<td>C (Cycle):</td>
<td>4 4</td>
<td>1 1</td>
</tr>
<tr>
<td>2V1 (2 Versus 1):</td>
<td>1 1</td>
<td>1 1</td>
</tr>
<tr>
<td>D (Diagonal):</td>
<td>1 1</td>
<td>1 1</td>
</tr>
<tr>
<td>13 (Tie from 1 to 3):</td>
<td>1 1</td>
<td>1 1</td>
</tr>
<tr>
<td>G3 (Group 3 different - disconnected):</td>
<td>1 1</td>
<td>1 1</td>
</tr>
<tr>
<td>10.2.2 Results</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As described above, in Stage 1 the simulations were performed using only networks with 11 vertices, partitioned into two or three clusters. In such small networks the individual blocks are very small and regular blocks are relatively dense, which makes it easier to discover the correct partition.
First, the results for the standard methods over all settings are presented. It should be emphasized that for settings with two-cluster partitions, exhaustive or full search\textsuperscript{106} for all generalized blockmodeling approaches. In settings with three-cluster partitions, a local search with 20 random starting partitions was used. In some settings with two-cluster partitions, a local search with 10 random starting partitions was used in addition to the full search. Although the results are not presented here, they were very similar to those obtained using a full search. However, this would probably not hold for larger networks.

The results using a local search for generalized blockmodeling approaches for $shape_1 = 8$ are presented in Figure 10.4 and for $shape_1 = 4$ in Figure 10.5, from which most of the conclusions can be drawn. The results for the other values of parameter $shape_1$ (10, 6, and 2) are presented in Appendix G.

In this thesis, figures are generally used to present the results of such simulations. As the number of different methods used is too large for all of them to be presented on a single graph, the methods are grouped and each group is presented in its own graph. Each figure is separated into several sections, with each hosting a graph. All these graphs have a common x axis, which is printed only once, at the bottom of the figure. The x axis contains the settings. The settings define the way that the networks in the simulations are generated. Those used in Figure 10.4 and Figure 10.5 are presented in Table 10.2. The label on the axis is comprised of the information in columns 2 to 5 of Table 10.2 (Table 10.4 for figures of Stage 2), separated by ‘|’. This is preceded by ‘T|’ if regularity was enforced, and by ‘F|’ if it was not, and followed by ‘|’ and the value of $shape_1$ parameter. On the y axis, the some statistic is usually represented. In most figures (including Figure 10.4 and Figure 10.5), this statistic is the mean of the Adjusted Rand Indices computed as described in Subsection 10.1.3.

In each section, there is a graph of the results for one group (blockmodeling type or group with some other common characteristics) of methods accompanied by a legend for these methods. Each small graph contains the background and main information. The background information comprises a background color and the thin lines. The legend for the background colors is found at the top of the figure. Each background color represents the different settings classes described in the previous subsection. The thin lines that are also part of the background provide information about the performance of methods from other groups. This is useful for positioning a group of methods that is in focus with the remaining methods based on their performance.

The main information is contained in the thicker lines, which are also the only lines for which the legend is provided. They represent the information about the performance (usually the

\textsuperscript{106} Exhaustive or full search means evaluating all possible partitions (computing criterion function) and selecting the partition with the smallest inconsistency. This is very time consuming, as the number of partitions is Stirling number of the second kind (Ferligoj, 1989: 58). This number rises very quickly with both the number of units and the number of clusters. E.g., there are 1023 partitions of 11 units into 2 clusters, 28501 partitions of 11 units into 3 clusters, about $16.8 \cdot 10^6$ partitions of 25 units into 2 clusters and about $1.4 \cdot 10^{11}$ partitions of 25 units into 3 clusters. Therefore, full search is not feasible in real applications.
Adjusted Rand Index) of the methods of the group that is in focus in a certain graph. The information is provided in the form of lines so it is easier to assess the performance of a method and to distinguish among methods. By no means is it meant as an indication that the x axis has a continuous scale since the settings are clearly measured on a nominal (discrete) scale.

The following conclusions can be drawn from the results in Figure 10.4, Figure 10.5 and Appendix G:

- **Binary blockmodeling:** In settings where valued networks are relatively close to binary ones \(^{107}\), binary blockmodeling performs very well, better than most other approaches or at least comparable to the better ones. These settings are those where the multiplication factors \((mf)\) are the same in all blocks and the \(shape2\) parameter has the value of 1 in all blocks. This holds in the case of reasonably large values of \(shape1\) and when regularity is enforced as the binary blockmodeling is most sensitive to the regularity being enforced (more than other methods). If the \(shape2\) and especially \(mf\) parameters differ across blocks, the binary blockmodeling is not an appropriate approach because it is usually one of the worst options. This applies to when pre-specified blockmodeling is not used. The conclusions based on the results of pre-specified blockmodeling are presented together for all approaches below.

- **Valued blockmodeling:** The expectation for valued blockmodeling was that it should perform at least as well as binary blockmodeling in all settings; however, such a performance was not achieved. It performed worse than binary blockmodeling in some settings where the valued network is relatively close to the binary one and the partition searched for is not maximal regular \(^{108}\). Even in most other settings, valued blockmodeling performed similarly to binary blockmodeling. It performed relatively consistently better than binary blockmodeling only in those settings where block-specific multiplication factors \((mf)\) (class of settings labeled Different block max.) are used.

- **Homogeneity blockmodeling:** Overall, these approaches, especially sum of squares blockmodeling, were the best approaches. Sum of squares blockmodeling performed well in all settings, with the exception of a few settings where regularity was not enforced, where it was nevertheless usually not much worse than the best method. Absolute deviations blockmodeling performed slightly worse, especially in those settings where the blockmodel was such that the partition searched for was not maximal regular and where no additional information about the block structure (such as different \(shape2\) parameters or multiplication factors \((mf)\) across blocks) was provided (class of settings labeled Not maximal regular). This holds true when the

---

\(^{107}\) E.g. settings ‘T|2| 1T| 1|10| 8’, ‘T|2| 13| 1|10| 8’, ‘T|2| BG| 1|10| 8’, ‘T|2| CP| 1|10| 8’ and ‘T|2| C| 1|10| 8’ in Figure 10.4.

\(^{108}\) E.g. settings ‘T|2| BG| 1|10| 8’, ‘T|2| CP| 1|10| 8’ and ‘T|2| C| 1|10| 8’ in Figure 10.4.
maximum was used as function \( f \) in \( f \)-regular blocks. When the mean was used as function \( f \), the results were much worse.

- **Implicit blockmodeling**: Implicit blockmodeling with only the regular block type allowed performed similarly as the homogeneity blockmodeling approaches, namely consistently well. When the null block type was also allowed, the performance slightly improved in some settings\(^{109}\). However, it was much worse in some other settings\(^{110}\).

- **REGE**: All versions of REGE performed relatively well in most settings. They only failed where the partition searched for was not maximal regular and where no additional information about the block structure (such as different shape parameters or multiplication factors \( mf \) across blocks) (class of settings labeled *Not maximal regular*) was provided (the setting T or F|2|AR|1|D, id 16 should also be included in that group as the block-specific multiplication factors \( mf \) are arranged in such a way that the partition being searched for is still not maximal regular even with this additional information). This cannot be seen as the bad performance of the algorithms as they are designed to find the maximal regular partition. Of the approaches used REGGE-OW (labeled ‘REGE.ow|reg’ in the figures) on average performed slightly better, which is the result of the fact that the definition of regular equivalence that it used is most similar to that used in the simulation of the networks. REGDI (labeled ‘REGD |reg’ in the figures), on the other hand, usually performed slightly worse than the rest. However, in general all four versions performed very similarly.

- **Pre-specified blockmodeling**: As was expected, pre-specifying the blockmodel substantially improved the results of binary, valued and implicit blockmodeling. Regardless of what was written before where individual blockmodeling types were discussed, binary, valued and implicit blockmodeling according to a pre-specified blockmodel were among the best approaches in almost all settings.

- **Structural equivalence**: Only two methods for structural equivalence were evaluated – sum of squares blockmodeling using complete blocks and an indirect approach (see Section 10.1.1 for details). Both methods performed unexpectedly well for these networks based on regular equivalence. They performed better than binary and valued blockmodeling (when pre-specified blockmodeling was not used) in most settings. They performed especially well when regularity was not enforced, where they were consistently very close to the best methods, if they themselves have not earned that title. The direct approach (sum of squares blockmodeling according to structural equivalence) performed consistently better than the indirect approach. In some settings\(^{111}\) the difference is quite substantial.

\(^{109}\) E.g. settings ‘T|3|G3|G2|10|8’, ‘T|3|2T|G3|10|8’ and ‘T|3|C2P|G3|10|8’ in Figure 10.4.

\(^{110}\) E.g. settings ‘T|2|CP|1|10|8’, ‘T|2|CP|4|10|8’ and ‘T|2|AR|D|10|8’ in Figure 10.4.

\(^{111}\) E.g. settings ‘T|2|IT|1|10|8’, ‘T|2|IT|4|10|8’, ‘T|2|BG|1|10|8’, ‘T|2|CP|1|10|8’ and ‘T|2|BG|1|R|8’ in Figure 10.4.
Figure 10.4: Results of simulations for all settings with shape1 = 8
Figure 10.5: Results of simulations for all settings with shape1 = 4
10.2.3 Effects of normalizations

In addition to the main body of simulations, a few additional simulations were also made to determine the effect of block size normalization (on all generalized blockmodeling types) and maximum normalization on implicit blockmodeling. These simulations were made using only a selection of settings (to save time) as indicated in Table 10.2. An attempt was made to select settings as diverse as possible, while not using those settings where the \(shape_2\) parameter was set to four in all blocks.

The results that show an effect of block size normalization are shown in Figure 10.6 for \(shape_1 = 8\) and in Figure 10.7 for \(shape_1 = 4\). Results for \(shape_1 = 10, 6\) and 2 are shown in Appendix H. In general, there is no consistent effect as the effect of the normalization can be either positive or negative. However, the effect is relatively consistent (when present) within certain settings (across different methods). For example, it is usually negative in settings \(2\mid T\mid 1\mid 10\) and \(2\mid CP\mid D\mid 10\), while it is usually positive in setting \(F\mid 2\mid AR\mid D\mid 10\).

The results that show the effect of maximum normalization are shown in Figure 10.8 for \(shape_1 = 8\) and in Figure 10.9 for \(shape_1 = 4\). Results for \(shape_1 = 10, 6\) and 2 are shown in Appendix I. Maximum normalization has a negative effect on the performance of the implicit blockmodeling in practically all settings where an effect occurs. For block size normalization, the conclusions are the same as those presented in the previous paragraph for all blockmodeling types. If both normalizations are used simultaneously, the effect is still negative.

Therefore, the use of maximum normalization is not advised. Regarding the block size normalization, no clear advice can be given based on these simulations. However, we must be aware that use of block size normalization has its pitfalls. It may lead to a blockmodel where one large block contains all the inconsistencies, while the remaining blocks are ideal (usually null, especially in sparse networks) blocks.
Figure 10.6: Effects of block size normalization for selected settings with shape1 = 8
Figure 10.7: Effects of block size normalization for selected settings with shape1 = 4

Table: Settings

<table>
<thead>
<tr>
<th>Clear pattern</th>
<th>Not maximal regular</th>
<th>Different dist. par.</th>
<th>Different block max.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

With block size normalization

Without block size normalization

Methods

- ss|str|sizeNorm
- ss|str
- sedist|str

- bin|reg|max|2min|sizeNorm
- bin|reg|max|max|sizeNorm
- bin|pre|max|2min|sizeNorm
- bin|pre|max|max|sizeNorm

- bin|reg|max
- bin|reg|max
- bin|pre|max
- bin|pre|max

- val|reg|max|2min|sizeNorm
- val|reg|max|max|sizeNorm
- val|pre|max|2min|sizeNorm
- val|pre|max|max|sizeNorm

- val|reg|max
- val|reg|max
- val|pre|max
- val|pre|max

- ad|reg|max|sizeNorm
- ss|reg|max|sizeNorm
- ad|reg|mean|sizeNorm
- ss|reg|mean|sizeNorm

- ad|reg|max
- ss|reg|max
- ad|reg|mean
- ss|reg|mean

- imp|reg|max|sizeNorm
- imp|wnull|reg|max|sizeNorm
- imp|pre|max|sizeNorm
- imp|pre|max|sizeNorm

- imp|reg|max
- imp|wnull|reg|max
- imp|pre|max
- imp|pre|max

- imp|reg|mean
- imp|pre|mean

T|2| 1T| 1|10| 4

F|2| CP| 1|10| 4

F|2| AR| 1| D| 4

F|2| CP| 1| R| 4

T|2| AR| 1| D| 4

F|2| CP| 1| R| 4

T|2| AR| 1| D| 4

F|2| CP| 1| R| 4

F|2| CP| 1| R| 4

F|2| CP| 1| R| 4

F|2| CP| 1| R| 4

F|2| CP| 1| R| 4

F|2| CP| 1| R| 4

F|2| CP| 1| R| 4

F|2| CP| 1| R| 4

F|2| CP| 1| R| 4

F|2| CP| 1| R| 4

F|2| CP| 1| R| 4
Figure 10.8: Effects of block size and maximum normalization (and their interaction) on implicit blockmodeling for selected settings with shape1 = 8
10.2.4 Effects of a different way of simulating the values of the ‘0’ ties

The ‘0’ ties in the network were generated from the beta distributions shown in Figure 10.2. However, this is often not a very realistic way of simulating the network. It is only realistic if we assume there is at least a very weak tie between each pair of units. If a tie between two units does not exist it is very unlikely to be reported. A more realistic approach in cases where every pair of units is not connected would be that a large percentage of the ‘0’ ties have the value of exactly 0. It is expected that most approaches will perform better in networks simulated in such ways. The improvement should be especially evident for binary and valued blockmodeling. These are additional simulations aimed at checking the affect of different ways of simulating the ‘0’ ties. An assumption is that the way ‘0’ ties are generated similarly affects the results regardless of the setting. Therefore, only three settings, those numbered 1, 3 and 10 in Table 10.2 with regularity enforced were used. These settings were chosen as the most classical representatives of the three (of the total four) classes of settings. These three classes are Clear pattern, Not maximal regular and Different dist. par. The fourth class of settings (Different block max.) was not chosen as it was deemed less realistic.

In this subsection the ‘0’ ties were generated in such as way that in 90% the ‘0’ tie had a value of exactly 0. A very large percentage of the ‘0’ ties that had a value of exactly 0 was selected
in an attempt to make the effect stronger. Values of the remaining 10% of the ‘0’ ties were simulated as before, based on the beta distributions presented in Figure 10.2.

The results are presented in Figure 10.10. When valued blockmodeling is used, this new way of simulating ‘0’ ties generally has a positive effect. It especially improves the performance of valued blockmodeling (at all values of \( \text{shape1} \) parameter), implicit blockmodeling with null blocks and binary blockmodeling (at lower values of \( \text{shape1} \) parameter). It usually also has a large positive effect on the performance of other methods at lower values of \( \text{shape1} \) parameter. However, it also has a negative effect in some cases. This negative effect is especially surprising for binary blockmodeling at larger values of the \( \text{shape1} \) parameter. It probably occurred as a result of the incorrect selection of the threshold \( t \) used in binarization of the network. We can therefore conclude that the way of generating the ‘0’ ties is partly responsible for the poor performance of binary and valued blockmodeling.

10.3 Stage 2

Stage 1 was meant to test the performance of different approaches to the blockmodeling of valued networks on a smaller scale, on networks with only 11 units. Therefore, it was possible to consider a wider range of settings and methods. For example, the effects of block size and maximum normalization were explored. In addition, due to a very limited number of different partitions of 11 units into two clusters, a full search was possible.

Stage 2, on the other hand, was designed to be more realistic since networks of 25 units were generated and analyzed. The majority of methods were applied to all settings, while pre-specified blockmodeling was only applied to a selection of settings. The number of repetitions (number of times a network for each setting was generated and analyzed with selected blockmodeling methods) was at least 20 for all settings; however, in many cases, more repetitions were made to generate more accurate estimates (e.g. of the mean Adjusted Rand Index). It can be seen from the results of Stage 1 that the relative performance of the methods is similar for different values of the \( \text{shape1} \) parameter. Therefore, the simulations were run using only two values of this parameter, 8 and 4.

10.3.1 Settings

As in Stage 1, several parameters were fixed during Stage 2 (used for generating networks in all the simulations in Stage 2). Some of them remained the same as in Stage 1, while others changed. The most important changes are the increase in the number of units to 25.

\[112\] Selection is done automatically based on the distribution of tie values.
Figure 10.10: Effects of a different way of simulating the ‘0’ ties
The basic characteristics of Stage 2 are:

- The number of units is 25
- Two different partitions were used:
  - a two-cluster partition:
    1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
  - a three-cluster partition:
    1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3

The two-cluster partition was used in most of the settings.
- The `shape1` parameter had values of 8 and 4.
- The parameter controlling the enforcement of strict regularity in the binary network (`enforce`) was set to TRUE and FALSE in most cases.
- The number of repetitions (`rep`) was usually set to at least 20. Based on each setting (combinations of parameters controlling the generation of networks), a network was generated at least 20 times. When possible, more simulations were allowed to more fully utilize the available computer time and obtain more accurate estimates (e.g. of the mean Adjusted Rand Index).
- A local search with 20 random starting partitions was used to find the partition with the lowest inconsistency in generalized blockmodeling. In a few selected settings, 100 random starting partitions were used to check if this would significantly improve the results.
- The number of iterations in REGE was set to 3.
- The methods used can be divided into two groups in terms of their use in Stage 2:
  - Common methods – methods used in all settings in Stage 2. These methods include both direct and indirect approaches. The indirect approaches are an approach for structural equivalence and the four versions of REGE. The direct approaches include several versions of binary, valued, implicit and homogeneity blockmodeling. These methods are marked by the following markings (explained in Subsection 10.1.1): `ss|str`, `sedist|str`, `bin|reg|halfmax`, `bin|reg|min`, `val|reg|max|2min`, `val|reg|max|max`, `imp|reg|max`, `imp|wnull|reg|max`, `ad|reg|mean`, `ad|reg|max`, `ss|reg|max`, `ss|reg|max`, `REGD|reg`, `REGD.ow|reg`, `REGE|reg`, and `REGE.ow|reg`.
  - Pre-specified blockmodeling – Pre-specified blockmodeling was used with binary, valued and implicit blockmodeling (markings: `bin|pre|halfmax`, `bin|pre|min`, `val|pre|max|max`, `val|pre|max|2min`, `imp|pre|max`) in only a subset of settings.
  - Pre-specified blockmodeling – Pre-specified blockmodeling was used with binary, valued and implicit blockmodeling (markings: `bin|pre|halfmax`, `bin|pre|min`, `val|pre|max|max`, `val|pre|max|2min`, `imp|pre|max`) in only a subset of settings.
- 100 starting partitions – in some settings (with `shape1 = 8` only) 100 starting partitions were used instead of the usual 20 to check if this could improve the partition. In these settings, the correct partition (the one used to generate the network) was also used (separately) as a starting partition to see if this partition would be recognized as the optimal partition, which means that no other
partition with lower inconsistency would be found by a local search. The inconsistency of the correct partition was also computed.

Other parameters used to generate the networks are shown in Table 10.4. Values in Table 10.4 (when not simple numbers for the shape2 and mf parameters) are explained in Table 10.5. When the value of the shape2 or mf parameter in Table 10.4 is a simple number (e.g. 1 or 4 for the shape2 parameter), it is the value of the parameter in all blocks (the number represents the matrix of appropriate size according to the number of clusters filled with this number).

Table 10.4: Settings used in the simulations – Stage 2

<table>
<thead>
<tr>
<th>Id</th>
<th>k</th>
<th>blockmodel*</th>
<th>shape2*</th>
<th>mf*</th>
<th>settings class</th>
<th>methods used</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>regularity enforced</td>
<td>regularity not enforced</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>CP</td>
<td>1</td>
<td>10</td>
<td>Not maximal regular</td>
<td>C, P, 100</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1T</td>
<td>1</td>
<td>10</td>
<td>Clear pattern</td>
<td>C, P, 100</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1T</td>
<td>4</td>
<td>10</td>
<td>Clear pattern</td>
<td>C, P, 100</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>BG</td>
<td>4</td>
<td>10</td>
<td>Not maximal regular</td>
<td>C, P, 100</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>BG</td>
<td>1</td>
<td>10</td>
<td>Not maximal regular</td>
<td>C, P, 100</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>CP</td>
<td>D</td>
<td>10</td>
<td>Different dist. par.</td>
<td>C, P</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>CP</td>
<td>R</td>
<td>10</td>
<td>Different dist. par.</td>
<td>C</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>BG</td>
<td>R</td>
<td>10</td>
<td>Different dist. par.</td>
<td>C, P</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>CP</td>
<td>1</td>
<td>D</td>
<td>Different block max.</td>
<td>C</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>BG</td>
<td>1</td>
<td>R</td>
<td>Different block max.</td>
<td>C, P</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>AR</td>
<td>1</td>
<td>D</td>
<td>Different block max.</td>
<td>C</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>AR</td>
<td>1</td>
<td>R</td>
<td>Different block max.</td>
<td>C</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>AR</td>
<td>D</td>
<td>10</td>
<td>Different dist. par.</td>
<td>C</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>AR</td>
<td>R</td>
<td>10</td>
<td>Different dist. par.</td>
<td>C</td>
</tr>
<tr>
<td>22</td>
<td>3</td>
<td>G3</td>
<td>G1</td>
<td>10</td>
<td>Different dist. par.</td>
<td>C</td>
</tr>
<tr>
<td>23</td>
<td>3</td>
<td>13</td>
<td>1</td>
<td>10</td>
<td>Clear pattern</td>
<td>C, P, 100</td>
</tr>
<tr>
<td>24</td>
<td>3</td>
<td>C</td>
<td>O4</td>
<td>10</td>
<td>Different dist. par.</td>
<td>C</td>
</tr>
<tr>
<td>25</td>
<td>3</td>
<td>C</td>
<td>1</td>
<td>10</td>
<td>Not maximal regular</td>
<td>C, P, 100</td>
</tr>
</tbody>
</table>

Legend:

k … the number of clusters
mf … multiplication factor
* the meaning of the values (in case of shape2 and mf only those that are not numbers) is explained in Table 10.3.
**Table 10.5: Values (codes) in Table 10.4 (for Stage 2)**

<table>
<thead>
<tr>
<th>Blockmodel</th>
<th>shape2 when block-specific</th>
<th>mf (multiplication factor) when block-specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP (Core-Periphery):</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[1,] &quot;reg&quot; &quot;reg&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2,] &quot;reg&quot; &quot;null&quot;</td>
<td></td>
</tr>
<tr>
<td>1T (1 Tie):</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[1,] &quot;null&quot; &quot;reg&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2,] &quot;null&quot; &quot;null&quot;</td>
<td></td>
</tr>
<tr>
<td>BG (Between Groups):</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[1,] &quot;null&quot; &quot;reg&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2,] &quot;reg&quot; &quot;null&quot;</td>
<td></td>
</tr>
<tr>
<td>G3 (Group 3 different):</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[1,] &quot;reg&quot; &quot;reg&quot; &quot;null&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2,] &quot;reg&quot; &quot;reg&quot; &quot;null&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[3,] &quot;null&quot; &quot;null&quot; &quot;reg&quot;</td>
<td></td>
</tr>
<tr>
<td>13 (Tie from 1 to 3):</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[1,] &quot;null&quot; &quot;null&quot; &quot;reg&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2,] &quot;null&quot; &quot;null&quot; &quot;null&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[3,] &quot;null&quot; &quot;null&quot; &quot;null&quot;</td>
<td></td>
</tr>
<tr>
<td>C (Cycle):</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[1,] &quot;null&quot; &quot;reg&quot; &quot;null&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[2,] &quot;null&quot; &quot;null&quot; &quot;reg&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[3,] &quot;reg&quot; &quot;null&quot; &quot;null&quot;</td>
<td></td>
</tr>
</tbody>
</table>

**10.3.2 Results for common methods**

The results for the common methods are presented in Figure 10.11 for \( \text{shape1} = 8 \) and in Figure 10.12 for \( \text{shape1} = 4 \). The number of repetitions (varying from 20 to 100) used to obtain a certain value of the Adjusted Rand Index (a point on a graph) is indicated by the size of the point. The size of the point increases with the logarithm of the number of repetitions used. The points in the graph have the same size as those in the legend when 20 repetitions were used. The results for pre-specified blockmodeling are presented in the next subsection.
Figure 10.11: Results of simulations in Stage 2 for all settings with shape1 = 8
Figure 10.12: Results of simulations in Stage 2 for all settings with shape1 = 4
In Appendix J the results are also presented in a tabular form with information on the value of the mean of the Adjusted Rand Index, the standard error of this mean and the number of repetitions (n) for all used combinations of different settings (including enforcing regularity or not and different shape\textit{l} parameters) and methods.

The clearest and also the most surprising result is the good performance of methods for structural equivalence. The conclusions based on the different groups of methods are:

- **Structural equivalence**: Methods for structural equivalence perform surprisingly well considering that the networks were generated based on regular equivalence. For example, they usually perform better than binary or valued blockmodeling (without pre-specified blockmodeling) for regular equivalence. However, they are usually not as good as methods of regular equivalence within homogeneity and implicit blockmodeling. When compared to REGE, they perform worse in settings where REGE performs well and better in most other settings. When comparing the two methods used for structural equivalence, it is clear that the direct approach performs usually much better than the indirect approach and never worse.

- **Binary and valued blockmodeling** (without pre-specified blockmodeling): Binary blockmodeling performed very badly in all settings. They are the worst methods in most settings. In most cases, valued blockmodeling performed only slightly better than binary blockmodeling. The most notable exceptions are the two-cluster settings from the class of settings Not maximal regular, where valued blockmodeling performed worse than binary blockmodeling and the settings with AR blockmodel and different block maximums by rows (of the blockmodel), where valued blockmodeling produced good results, while binary blockmodeling performed poorly as in other settings. Some ideas for the bad performance of these methods are presented at the end of this subsection.

- **Homogeneity blockmodeling**: These methods (those based on \textit{max}-regular equivalence) were overall the best methods used and were the best methods in most settings. They were significantly outperformed by some other methods only in the class of settings Not maximal regular. In these settings, the sum of squares approach for structural equivalence performed the best. The measure of variability used (sum of squares versus absolute deviations) did not have an effect as they both performed similarly. The homogeneity blockmodeling based on \textit{mean}-regular equivalence performed much worse in all settings.

- **Implicit blockmodeling** (without pre-specified blockmodeling): Implicit blockmodeling with a regular block only performed similarly to homogeneity blockmodeling according to \textit{max}-regular equivalence, although slightly worse. However, it also performed slightly better in some settings. Unlike in Stage 1, using null blocks in addition to regular blocks led to considerably inferior results in most settings. When using null blocks, the results were similar to those of valued blockmodeling.
- **REGE**: REGE performed relatively well in most settings where the partitions that were searched for were maximal regular at least when valued information is taken into account (all settings except those labeled ‘not maximal regular’ and setting ‘(T/F) | 2 | AR | 1 | D’). This is understandable as REGE was designed to find the maximal regular partition. However, even in such settings it usually performed slightly worse than homogeneity and implicit blockmodeling, especially when the \textit{shape} parameter was 8 and regularity was not enforced. All four versions performed similarly in most cases. Still, REGE.ow (REGGE-OW) performed on average slightly better than the rest as it is also theoretically the most suitable REGE version. That is, the networks that were generated based on partitions are more similar to the ideal networks based on these partitions of that REGE algorithm than to any other (REGE algorithm).

Two of the results presented above are most surprising. The first one is the good performance of sum of squares blockmodeling according to structural equivalence. The second is the bad performance of binary and valued blockmodeling when using pre-specified blockmodels.

One possible reason for the good performance of sum of squares blockmodeling lies in the fact that sum of squares blockmodeling does not compare individual pairs of ties as indirect approaches or searches for blocks where all tie values are either approximately zero or over some pre-specified value (as values and binary blockmodeling). Sum of squares blockmodeling instead searches for blocks that are relatively homogeneous, where cell values are as close to the mean of the cell values in that block. Therefore, it tries to cluster most of the high values together and most of the low values together. First, this allows it to identify null blocks. Second, as it is usually impossible to find blocks with only high values the next best thing to do is to find blocks where there is larger concentration of high values than in other blocks. Sum of squares blockmodeling according to structural equivalence can do that and these blocks are usually regular blocks that are searched for or at least similar to them.

The poor performance of binary and valued blockmodeling, especially in the class of settings Clear pattern and partly Not maximal regular when regularity was enforced, is very surprising. Several factors may have contributed to such results. Results of the evaluation of the optimization presented later in Subsection 10.3.4 indicate at least two possible factors. The first, that affects mainly binary blockmodeling especially in the Not maximal regular class of settings when regularity is enforced, is that binary blockmodeling simply does not measure the inconsistency of a partition with ‘valued’ regular equivalence, that is in valued networks, precisely enough. This is indicated by the fact that partitions other than the correct partition and in fact very different from the correct one with no inconsistencies (with the value of the criterion function or total inconsistency of the partition equal to 0). The applications of binary blockmodeling to real networks (in Chapter 9 and in general) have also shown its tendency to find several partitions as ‘optimal’ when used to find relatively ‘loose’

\footnote{113 What kind of network is ‘ideal’ for most a versions of REGE algorithms is demonstrated in Section 9.2.}
equivalences such as regular equivalence. The other factor that was based on Subsection 10.3.4 that is mostly present in the class of settings Clear pattern when regularity is enforced (which are most suited out of all the settings for both binary and valued blockmodeling) is that the optimization procedure. Figure 10.16 shows that in these settings the correct partition has a smaller value of the total inconsistency (criterion function) than the partition found using the local optimization of 20 (or even 100) random starting partition. Obviously, the optimization procedure only finds the local and not global minimums of the criterion function (total inconsistency).

10.3.3 Results for pre-specified blockmodeling

The results for pre-specified blockmodeling are presented in Figure 10.13 for \( shape1 = 8 \) and in Figure 10.14 for \( shape1 = 4 \). As indicated in Table 10.4, pre-specified blockmodeling was used in only some settings in order to save time. Those on which it was used were selected as representatives of appropriate classes of settings. More settings were selected from the setting classes Clear pattern and Not maximal regular as these were the two classes of settings where the poor performance of binary and valued blockmodeling was the most surprising. This also made sense for implicit blockmodeling as, among the settings where regularity was enforced, these were the only classes of settings where implicit blockmodeling had much room for improvement.

As expected, pre-specified blockmodeling improved the performance of all blockmodeling types where it was used (in the case of implicit blockmodeling only if we compare it to its use with both null and regular blocks). The binary and valued blockmodeling according to pre-specified blockmodel performed similarly, although valued blockmodeling (with parameter \( m \) determined as the maximum tie value in the network) performed slightly better, especially in settings where \( shape1 = 4 \). Implicit blockmodeling according to a pre-specified blockmodel performed worse than the binary and valued blockmodeling according to pre-specified blockmodel. The methods according to pre-specified blockmodel especially excelled in the class of settings Not maximal regular, where they outperformed all other methods. Even in other settings, these blockmodeling types with pre-specified blockmodeling performed reasonably well, especially when \( shape1 = 8 \), particularly when compared to the terrible performance of binary and valued blockmodeling without pre-specified blockmodeling.
Figure 10.13: Results for pre-specified blockmodeling for selected settings where parameter shape1 = 8
10.3.4 Evaluation of optimization

The results obtained using the direct approach presented up till now in this section were produced using the local optimization of 20 random starting partitions. The aim of this subsection is to try to evaluate if the poor performance (when it occurred), especially the very poor performance of binary and valued blockmodeling, might be caused by the fact that the local optimization of 20 random starting partitions finds only locally but not globally optimal partitions. The aim is also to assess if the correct partitions could be globally optimal partitions. This is done only on a subset of settings (also indicated in Table 10.4) in order to save time. Only settings from two classes of settings were selected, that is from the classes Clear pattern and Not maximal regular, as these were the two classes of settings where the
poor performance of binary and valued blockmodeling was the most surprising. These are also the two classes of settings where other approaches have most room for improvement.

In order to achieve this, the Adjusted Rand Indices are compared with those obtained using the local optimization of 100 starting partitions and with those obtained with an optimization of the correct partition (the one used in generation of the networks). The comparison is only made on a few selected settings as mentioned in Table 10.4. The results are presented in Figure 10.15. However, here the comparison is not made only according to the Adjusted Rand Indices but also to the (total) inconsistencies\textsuperscript{114}. The inconsistencies obtained using the local optimization of 20 random starting partitions are compared with those obtained using the local optimization of 100 random starting partitions, with those obtained using the optimization of the correct partitions, and with the inconsistencies of the correct partitions. The values of the inconsistencies are presented in Figure 10.16. The conclusions based on both Adjusted Rand Indices and the inconsistencies are:

- **Structural equivalence**: Using more random starting partitions did not have a consistent effect on sum of squares blockmodeling according to structural equivalence. When only using the correct partition as the starting partition, the obtained partition was closer to the correct partition which is, of course, natural as it served as a starting partition. Especially striking is the similarity of the inconsistencies obtained with all four methods and that the inconsistencies of the correct partitions are almost the same as those obtained using some form of local optimization that nevertheless produced quite different partitions. Because of such similarity of the obtained inconsistencies I was concerned that an error had been made when producing these figures. However, a careful examination of the generated networks confirmed that all these relatively different partitions have very similar inconsistencies, which are obviously also very close to the global maximum, as otherwise it would be expected that the inconsistencies obtained with 100 random starting partitions would be considerably smaller.

- **Binary blockmodeling** (without pre-specified blockmodeling): Using 100 random starting partitions instead of 20 improved the performance of binary blockmodeling in two settings (although not enormously), while not having much effect in the other settings. Using the correct partition as a starting partition led to very good results, especially in those settings where regularity was enforced. Settings where regularity was enforced using the correct partition as a starting partition also led to smaller inconsistencies, indicating that the convergence to a global (and not a local) maximum is a problem. In settings where regularity was not enforced, these approaches led to higher inconsistencies and the correct partition was associated with even higher ones, indicating that binary blockmodeling is inappropriate for such networks.

\textsuperscript{114} Total inconsistency measures how inconsistent are all empirical blocks with the ideal blocks. It represents the valued or the criterion function that is minimized in direct blockmodeling approaches.
• **Valued blockmodeling** (without pre-specified blockmodeling): Using 100 random starting partitions instead of 20 did not consistently improve the performance of valued blockmodeling. On the other hand, using the correct partition as a starting partition of the local optimization had a similarly favorable effect as on binary blockmodeling. However, it did not consistently lower the inconsistencies as was the case with binary blockmodeling, indicating that it just led to more similar local optima. Also, the inconsistencies of the correct partitions were not consistently lower than of those obtained with a local optimization with random starting partitions, indicating that it might not be an appropriate approach to such problems.

• **Homogeneity blockmodeling:** Increasing the number of random starting partitions considerably improved the performance of sum of squares blockmodeling according to max-regular equivalence in the class of settings Not maximal regular (especially with regularity enforced), while it had little effect in other settings and methods. Using the correct partition considerably improved the results. However, it also increased the inconsistencies in some settings, although considerably in just two settings. The inconsistencies of the correct partition were similar or slightly higher.

• **Implicit blockmodeling** (without pre-specified blockmodeling): For implicit blockmodeling very similar things could be stated as for homogeneity blockmodeling, except the effect of the increased number of random starting partitions is even smaller. For most of the methods, the increase in the number of random starting partitions did not have a considerable effect although there are some combinations of settings and methods where an improvement was noted. Using the correct partition almost always led to higher Adjusted Rand Indices as could expected since the same partition used in comparison was also used as a starting partition. However, in most cases use of the correct partition as a starting partition often led to higher inconsistencies, with binary blockmodeling (when regularity was enforced) being the most notable exception. The inconsistencies for the correct partition were also often higher, with binary blockmodeling (when regularity was enforced) again being the most notable exception. The inconsistencies of the correct partition higher than the partitions obtained through local optimization are especially problematic when the Adjusted Rand Indices of the partitions obtained through local optimization are relatively low. In such situations, we can conclude that those approaches where this occurs are inappropriate for partitioning the networks generated in these simulations.
Figure 10.15: Results obtained by a local optimization of 20 and 100 random starting points and of the correct partition.
Figure 10.16: Comparison of inconsistencies

<table>
<thead>
<tr>
<th>Settings</th>
<th>Clear pattern</th>
<th>Not maximal regular</th>
<th>Different dist. par.</th>
<th>Different block max.</th>
</tr>
</thead>
</table>

Comparison with:
- opt
- of
- 100
- sta
- rtin
- poi
- nts

Methods
- bin|reg|halfmax
- bin|reg|max
- bin|reg|halfmax|100
- bin|reg|min|100
- bin|reg|halfmax|OC
- bin|reg|min|OC
- bin|reg|halfmax|Cor
- bin|reg|min|Cor

Methods
- val|reg|max|2min
- val|reg|max|max
- val|reg|max|2min|100
- val|reg|max|max|100
- val|reg|max|2min|OC
- val|reg|max|max|OC
- val|reg|max|Cor
- val|reg|max|Cor

Methods
- ad|reg|max
- ss|reg|max
- ad|reg|max|100
- ss|reg|max|100
- ad|reg|max|OC
- ss|reg|max|OC
- ad|reg|max|Cor
- ss|reg|max|Cor

Methods
- imp|reg|max
- imp|wnull|reg|max
- imp|reg|max|100
- imp|wnull|reg|max|100
- imp|reg|max|OC
- imp|wnull|reg|max|OC
- imp|reg|max|Cor
- imp|wnull|reg|max|Cor
10.4 Generalized equivalence

In addition to the simulations of regular networks, two settings based on generalized equivalence were also used in the simulations. In these two settings, the number of units was set to 15, the number of clusters to 3, shape2 parameter to 1, multiplication factor (mf) to 1, regularity was always enforced, and only two values (4 and 8) were used as shape1 values. The following blockmodels were used:

R (Row):

<table>
<thead>
<tr>
<th>[1,]</th>
<th>[2,]</th>
<th>[3,]</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;reg&quot;</td>
<td>&quot;null&quot;</td>
<td>&quot;null&quot;</td>
</tr>
<tr>
<td>&quot;cdo&quot;</td>
<td>&quot;null&quot;</td>
<td>&quot;null&quot;</td>
</tr>
<tr>
<td>&quot;cdo&quot;</td>
<td>&quot;null&quot;</td>
<td>&quot;null&quot;</td>
</tr>
</tbody>
</table>

B (Both row and column):

<table>
<thead>
<tr>
<th>[1,]</th>
<th>[2,]</th>
<th>[3,]</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;reg&quot;</td>
<td>&quot;rdo&quot;</td>
<td>&quot;rdo&quot;</td>
</tr>
<tr>
<td>&quot;cdo&quot;</td>
<td>&quot;null&quot;</td>
<td>&quot;null&quot;</td>
</tr>
<tr>
<td>&quot;cdo&quot;</td>
<td>&quot;null&quot;</td>
<td>&quot;null&quot;</td>
</tr>
</tbody>
</table>

The approaches used were those used in Stages 1 and 2 with mainly the same options. However, the equivalences were specified differently. For each blockmodeling type, the following allowed block types were used as a definition of equivalence (they are preceded by the labels used in Figure 10.17):

- **str**: null and complete blocks (structural equivalence)
- **reg**: null and regular blocks (regular equivalence)
- **dom**: null, regular, row-dominant, and column-dominant blocks
- **pre**: pre-specified blockmodeling

When using homogeneity blockmodeling, the null block type was not allowed except in pre-specified blockmodeling. Version 3 of the row- and column-dominant blocks was used (in homogeneity blockmodeling).

The results in Figure 10.17 show that when only using blocks corresponding to structural or regular equivalence, the direct approaches did not perform very differently (at least systemically) than the indirect ones. However, when row- and column-dominant blocks and even more when pre-specified blockmodeling was used all direct approaches clearly outperformed the indirect ones (and themselves using structural or regular equivalence) when the B blockmodel is used, while for blockmodel R this is true only for binary and only in the case of pre-specified blockmodeling also for valued blockmodeling. For homogeneity and implicit blockmodeling, including dominant blocks even decreases the performance when blockmodel R is used. For homogeneity blockmodeling, it should be kept in mind that the definitions of row- and column-dominant blocks are experimental. Here, further simulations and research are obviously required.
10.5 Limitations

The simulation study presented in this chapter has several limitations. These limitations are either due to the way the networks were generated or due to the implementation of the methods used to analyze these networks.

The following aspects of the way the networks were simulated might have made the generated networks less realistic and thus rendered the results less relevant:

- **Simulation of the ‘0’ ties:** The ‘0’ ties were simulated from the beta distribution described in Subsection 10.1.2. This might be problematic since practically none of these ‘0’ ties had an exact value of 0. A tie value of 0 means that the tie does not exist. Most real networks are, on the other hand, usually relatively sparse, that is, they have a lot of pairs of units that are not
connected, they have a lot of ties with value of exactly 0. There are also examples of real networks where exact ‘0’ ties are rare, e.g. trade networks among countries. The effect of this assumption was tested in Subsection 10.2.4 where it was shown that giving the exact value of 0 to 90% of the ‘0’ ties improves the performance in most combinations of settings and methods. However, the effect is not strong in those settings with a high shape parameter (except for valued blockmodeling).

- **no binary inconsistencies in null blocks**: The binary networks based on which valued networks were obtained were generated without any inconsistencies in null blocks. The assumption was that ties are rarely reported if not preset and that the inconsistencies based on values added to these ‘0’ ties would be sufficient. The problem with this assumption is that it contradicts the idea that at least a large portion of the ‘0’ ties does not have to have a value of exactly 0.

- **distribution used for generating the tie values**: In the simulation, a beta distribution with specific parameters was used to generate the tie values. Although the choice of the distribution and the parameters was based on real networks, they do not represent all networks. For certain types of networks, different distributions or parameters are more appropriate. Also, the values in most of the real networks analyzed in Chapter 9 had a discrete tie value distribution, while here a continuous distribution was used.

The implementation of the methods used in these simulations also affected the results. Therefore, the results can be used to compare the implementation of these methods and not the methods themselves. This problem is especially relevant for generalized blockmodeling methods. On the other hand, the implementation of all generalized blockmodeling approaches suffers from the same problems listed below, which means that the results obtained in this chapter can be used to compare them. The most problematic aspects of implementing the generalized blockmodeling approaches in the blockmodeling 0.1.2 package (Žiberna, 2006) are:

- **the experimental state of the package**: the blockmodeling package is still experimental. Most of the following limitations stem from this fact, e.g. little effort was put into optimizing the code for speed, some procedures are still being improved etc.

- **speed of execution**: one of the main problems of implementing the generalized blockmodeling approaches used in this chapter is the speed of execution. The most serious affect this limitation of the implementation had on these simulations is that usually local optimization was done on only 20 random stating partitions. This is not only a problem of the methods (which are by themselves very computationally intensive), but also of the implementation since the implementation of binary blockmodeling in Pajek 1.11 (Batagelj and
Mrvar, 2006) is about 1000 times faster than the one in the package used. If 1000 random starting partitions could be optimized, we could be much more confident that the partition(s) found using a local optimization is at least close to the global optimum. However, this was not possible as even by optimizing just 20 random points, about two weeks and 40 computers were needed to complete the simulations.

- **generation of the random starting partitions:** One thing that was not implemented well in version 0.1.2 is the procedure used to generate starting partitions for the local optimization (used in generalized blockmodeling approaches). In this version, the starting partitions were chosen so the probabilities of belonging to a cluster are the same for all clusters. Such a procedure has a tendency to generate partitions with approximately equally sized clusters. In real applications, this is a serious shortcoming as the partitions are often not even approximately equally sized. Although the partitions used to generate networks in this chapter did not have clusters of extremely different sizes, this shortcoming might still have serious consequences in the evaluation of generated networks since partitions with unequal clusters may be a more suitable starting partition. In any case, the same procedure was used to select the starting partitions for all generalized blockmodeling approaches.

## 10.6 Conclusions

These conclusions are mainly based on the results of Stage 2. The reason for that is that Stage 1 was chiefly intended as a preliminary stage. Networks with only 11 units were analyzed in Stage 1, which is not enough units to produce relevant results regarding regular equivalence. The difference between the results of Stages 1 and 2 is striking. Most methods, but especially binary and valued blockmodeling, performed considerably better in Stage 1. Two factors are most likely responsible for this. The first one is that the generated regular blocks are much denser when they are smaller and denser blocks are easier to identify. Since the number of units is smaller in Stage 1 so are the clusters and consequently the blocks. The second one is there are much less possible partitions into two or three clusters of 11 units (the number of units used in Stage 1) than of 25 (the number of units used in Stage 2). Due to this, a full search was used in Stage 1 to find two-cluster partitions. For three-cluster partitions a local optimization with 20 random starting points was used, however this was sufficient for such a small number of units. In contrast, 20 random starting partitions was shown to be inadequate (Subsection 10.3.4) for partitions of 25 units into two or three clusters.

The most surprising result is the relative effectiveness of the methods for structural equivalence on simulated networks based on max-regular equivalence. The sum of squares blockmodeling according to structural equivalence (direct approach) performed especially well. Although this clearly also proves the usefulness of these methods for the analysis of such networks, it also creates doubt about whether the way the networks were generated was
appropriate. The doubt that the effectiveness of the methods for structural equivalence is partly due to a suboptimal simulation setting is also supported by some examples (Section 9) in this thesis since in most examples the methods for structural equivalence often did not lead to satisfactory results. There are however some arguments that explain the good performance of sum of squares blockmodeling on regular networks, which were explained near the end of Subsection 10.3.2. Therefore, we can conclude that the good performance of methods for structural equivalence might be the result of the way the networks were generated and should be taken with caution, but not dismissed. Nevertheless, the simulations produced valid and useful results.

When comparing the two methods for structural equivalence, namely an indirect approach for structural equivalence or, more precisely, Ward’s hierarchical clustering of distances computed using Corrected Euclidean-like dissimilarity (Burt and Minor, 1983 in Batagelj et al., 1992: 71) and a direct approach of sum of squares blockmodeling according to structural equivalence, the second performed much better.

The simulation results have also shown that the methods developed in this dissertation, especially homogeneity and implicit blockmodeling, are better tools for identifying regular partitions in a valued network than the existing direct methods. Until now, the REGE algorithm (the direct approach) was the only method designed to find partitions in terms of regular equivalence in valued networks. The homogeneity blockmodeling methods (with max-regular blocks) have in particular in almost all settings performed better or at least as well as REGE. In addition, homogeneity blockmodeling like other methods of generalized blockmodeling can, unlike REGE, also be used when different kinds of equivalences (other than regular) are assumed to partition the network. It can be used with other allowed block types (other than (max-)regular) and to some extent also with pre-specified blockmodels. REGE performed well in all settings for which it was designed, that is in settings where the partition that was searched for was maximal regular even when the tie values are not taken into account. It was expected that it would also perform well in settings where the partition that was searched for could be considered maximal regular only if the values were also taken into account. Although it also performed relatively well in those settings, its performance was not as good as that of homogeneity and implicit blockmodeling, especially in settings where regular blocks differed (which made the partition that was searched for maximal regular) due to the use of different parameters in the beta distribution used to simulate the tie values. Implicit blockmodeling also performed similarly to homogeneity blockmodeling (with max-regular blocks).

Binary and valued blockmodeling performed relatively well in Stage 1, especially when the partition searched for was not maximal regular and no additional information in terms of tie values was provided. Unfortunately, they usually performed terribly in Stage 2 (where a local search with 20 random starting partitions was used in networks with 25 vertices). Especially in the case of binary blockmodeling, there are indications that the problem is in the search procedure as the correct partition usually had less inconsistency than the one found by a local search. As mentioned at the beginning of this section, the number of random stating partitions
used (20) is too small for a network with 25 units. Also, the random stating partitions might not have been chosen optimally. Both points are discussed in the previous section.

However, the poor performance only occurs when these two blockmodeling types were not used with pre-specified blockmodeling. When this additional information was used, they were usually the best or among the best methods. The use of pre-specified blockmodeling also considerably improved the results of implicit blockmodeling in settings where the partition that was searched for was not maximal regular.

When simulating networks based on the generalized equivalence, the generalized blockmodeling types showed their potential, although some additional research is clearly needed at least for homogeneity blockmodeling.

10.7 Ideas for future work

Many questions are still open and could be answered using simulations. One of the more technical ones is what is an appropriate way of generating networks (especially '0' ties).

There is also a need for further simulation whereby blockmodeling methods could be evaluated more thoroughly for networks generated based on generalized equivalence. These networks should be generated based on blockmodels with mixed types of blocks, which are blockmodels that include the block of type null and of at least two block types among the following: regular, row- and column-dominant, row- and column-functional etc.

Also, two of the open problem sets mentioned by Doreian (2006: 126) are very appropriate to be studied by simulations. These are the effects of missing data (or more generally inaccurate data) and of the network boundary problem. The network boundary problem arises when it is problematic to determine which units should be included in the network.

In addition to extending the simulations by including networks with different features as suggested above, it would also be useful to evaluate other methods such as stochastic blockmodeling and other optimization procedures (within generalized blockmodeling) in comparison to a local search etc.
11 CONCLUSIONS

After a short overview of the dissertation, a comparison and evaluation of the approaches presented in this dissertation is given. Then, some suggestions are made regarding the use of the proposed (and some other) approaches to blockmodeling. The chapter concludes with some ideas for the future research on blockmodeling of valued networks.

11.1 A short overview

The main purpose of this dissertation is to present new approaches to the generalized blockmodeling of valued networks. Three such approaches were developed in this dissertation, valued blockmodeling, homogeneity blockmodeling and implicit blockmodeling. The last one is heavily based on the ideas of Batagelj and Ferligoj (2000: 12-13) and those on the previously mentioned approaches.

To put these approaches into perspective, networks, blockmodeling and equivalences were first reviewed in Chapters 2 and 3. The most important contribution of these two chapters is their discussion of equivalences for a valued network, where also a new equivalence was proposed. This equivalence is called $f$-regular equivalence and can be thought of as an extension of regular equivalence to valued networks.

Indirect approaches to blockmodeling were then studied in Chapter 4. Several modifications were suggested for one of the indirect approaches, namely for REGE (White, 2005). These modifications aimed to make this approach more similar (or compatible) to generalized blockmodeling and to make it more suitable for the type of networks where tie values are influenced by the size of the units (e.g., trade networks, networks of carbon flow among species).

The direct approach was introduced in Chapter 5. Generalized blockmodeling, which is the most relevant representative of the direct approach for the dissertation was introduced in Chapter 6. The binary (generalized) blockmodeling (Doreian et al.: 2005) presented there, its type of criterion function and its ideal blocks were used as the basis on which the approaches to the generalized blockmodeling of valued networks were developed in Chapter 7.

The first proposed approach to blockmodeling of valued networks, valued blockmodeling, is a straightforward generalization of binary blockmodeling (Doreian et al., 2005). It was shown that all the features of binary blockmodeling can be generalized to valued blockmodeling\textsuperscript{115}. Binary blockmodeling is in fact only a special case of valued blockmodeling. Therefore, it was expected that for a valued network where the measurement scale is at least interval, the valued blockmodeling would always be superior or at least equal to binary blockmodeling.

\textsuperscript{115} Generalization to two-mode networks, other block types etc.
This proved to be true in all examples in Chapter 9 and in most but not all the simulations presented in Chapter 10.

We can say that binary blockmodeling treats ties either as relevant or as nonexistent (or irrelevant). Valued blockmodeling is similar, yet it assesses if a tie is closer to being relevant or closer to nonexistent. If this is all that is important and if we have an idea about an appropriate criterion for determining which ties are relevant, these two approaches are well suited for partitioning the network and determining types of blocks in the obtained blockmodel. They can, however, not distinguish between blocks that do not differ in the type of block but only in the values of ties relevant for that block type. Consequently, such differences in tie values cannot be used to find partitions.

The second presented approach is the homogeneity approach. It addresses one of the main problems of valued and binary blockmodeling. When using valued blockmodeling, a threshold must be explicitly selected that tells us how strong a tie must be to be treated as relevant. A similar threshold must also be selected when using binary blockmodeling, although it is sometimes implicitly set to the minimum positive value (all ties that exist, meaning those higher than 0, are treated as relevant). However, the compatibility of some block types in homogeneity blockmodeling is questionable. In addition, the null block type is not really used in homogeneity blockmodeling (without some modifications) as it is seen as a special case of a complete block type. Therefore, homogeneity blockmodeling is well suited for distinguishing empirical blocks based on tie values and finding partitions based on such differences, but it is not suited for distinguishing empirical blocks based on block types and finding partitions based on such distinctions. This is especially problematic since it is also not well suited for distinguishing between the null block type and other block types, a distinction that is of great importance in blockmodeling. The null block type is namely the only block type that does not induce a tie between equivalence classes (or clusters) in the reduced graph.

The third approach, implicit blockmodeling, was developed based on the ideas of Batagelj and Ferligoj (2000, 11-13) presented at the end of Chapter 6. Implicit blockmodeling can distinguish empirical blocks based on both tie values and block types. However, due to the fact that it is heavily influenced by the block maximums it often classifies blocks differently to what would be desired. As the partitions that it finds are heavily influenced by the classification of blocks, it often leads to unsatisfactory partitions. This is especially problematic if block maximum normalization is also used. The partitions that it produces can be improved, but this improvement comes at the price of one of the main disadvantages of implicit blockmodeling, namely its ability to distinguish between the null block type and other block types. Nevertheless, some examples in Chapter 9 and the simulations in Chapter 10 showed that implicit blockmodeling can produce good results. The approach is especially not suitable for use in networks where the distribution of the tie values has a long right tail.

All the suggested approaches are implemented in a blockmodeling package (Žiberna, 2007a) for statistical program and environment R (R Development Core Team 2006). They and some other approaches are then tested on four empirical and two constructed examples in
Chapter 9. The emphasis is usually on the generalized blockmodeling approaches, especially those for the blockmodeling of valued networks. The exceptions are two examples where different versions of the REGE algorithm are examined. Several properties or characteristics of the approaches or their implementations were learned or confirmed through these examples. The results that follow are mainly based on the empirical examples.

One problem any researcher, attempting to use blockmodeling approaches in real networks, will face is which approach is the most suitable. Ideally, researchers would select the approach whose definition of some equivalence or ideal blocks best matches the theoretical problem they want to solve by obtaining partitions. However, this is rarely possible in a more exploratory analysis.

Broadly speaking, most generalized blockmodeling usually performed well, although binary blockmodeling sometimes provided less suitable partitions or a larger number of partitions (even several hundreds of partitions were found to be optimal based on the values of criterion function). One of the valued blockmodeling partitions was often selected as the most suitable one, but this might also be partly due to the fact that several values of parameter \( m \) were used and the best partitions were then selected. Currently, binary blockmodeling has a unique advantage in terms of implementation as it is the only approach implemented in a relatively fast program (in Pajek 1.11 (Batagelj and Mrvar, 2006)).

Homogeneity blockmodeling does not suffer these problems, but from the so-called ‘null block problem’. This brings two undesirable effects that were observed in the examples. The first is that homogeneity blockmodeling by itself cannot classify blocks as null as opposed to some other block type. The second effect is that homogeneity blockmodeling often produces less clear ‘null blocks’.

Implicit blockmodeling produces good partitions if it is used without null blocks. If used in this way, ‘the null block problem’ also applies to implicit blockmodeling although the effect, whereby the null blocks are less clear, is not as serious. When used with null blocks, the partitions obtained are usually slightly worse and the blockmodels are inappropriate.

The indirect approaches perform relatively well in most examples, although usually not as well as the generalized blockmodeling approaches. In some cases, the indirect approaches are able to identify reasonable partitions at some, but not at all numbers of clusters in a partition. Due to the speed of their calculations, their use is strongly supported at least as a preliminary step (to assess the appropriateness of the structural or regular partitions and of the number of clusters).

As mentioned, generalized blockmodeling usually performed better. However, for networks where the tie values that are in some sense influenced by the ‘size’ of the units (e.g., trade networks among countries, carbon flow networks), a version of REGE especially designed for

---

116 As mentioned above, homogeneity blockmodeling does not classify blocks as null. The classification is usually done on block averages or some similar statistics based on tie values in a block.
such networks provides good partitions. Generalized blockmodeling might provide reasonable partitions if applied to a suitably normalized network, but they were unable to provide such a partition when simultaneously used on row and column normalized matrices.

In two examples image graphs are constructed based on the partitions obtained. Several rules are used for obtaining these image graphs. The averaging rules based on sum of squares blockmodeling and valued blockmodeling probably produce the best image graphs. Those based on the sum of squares approach give more emphasis to the tie values in the block (while taking into account the type of block), while those based on valued blockmodeling give more emphasis to the pattern of ties (while taking into account how far the tie values are from 0 and from the parameter m).

A relatively comprehensive simulations study where the proposed approaches are evaluated on generated valued regular networks was presented in Chapter 10. The simulations were carried out in two stages. These conclusions are mainly based on the results of Stage 2. The reason for that is that Stage 1 was primarily intended as a preliminary stage as networks with only 11 units were analyzed in Stage 1. Only 11 units are not enough to produce relevant results regarding regular equivalence.

The most surprising result is the relative effectiveness of the methods for structural equivalence on simulated networks based on \( \text{max} \)-regular equivalence. Sum of squares blockmodeling according to structural equivalence (a direct approach) performed especially well. They usually did not perform as well as methods for regular equivalence, but much better than expected. When comparing the two approaches to blockmodeling according to structural equivalence, namely an indirect and a direct approach, the second performed much better.

The simulation results also showed that the methods developed in this dissertation, especially homogeneity and implicit blockmodeling, are better tools for identifying regular partitions in a valued network than the existing direct and indirect methods. REGE performed well in all settings for which it was designed, that is in settings where the partition that was searched for was maximal regular, although the results were not as good if the partition that was searched for could be considered maximal regular only if the values were also taken into account. Implicit and especially homogeneity (with \( \text{max} \)-regular blocks) blockmodeling were generally the best approaches.

Binary and valued blockmodeling performed relatively well in Stage 1, especially when the partition searched for was not maximal regular and no additional information in terms of tie values was provided. Unfortunately, they usually performed terribly in Stage 2. Especially in the case of binary blockmodeling, there are indications that the problem lies in the search procedure as the correct partition usually had less inconsistency than that found by a local search.

However, the poor performance only occurred when these two blockmodeling types were not used with pre-specified blockmodeling. When this additional information was used, they were usually the best or one of the best methods. The use of pre-specified blockmodeling also
considerably improved the results of implicit blockmodeling in settings where the partition that was searched for was not maximal regular.

When simulating networks based on generalized equivalence, the generalized blockmodeling types showed their potential although some additional research is clearly needed at least for homogeneity blockmodeling.

All these approaches were also implemented in an R (R Development Core Team, 2006) blockmodeling package (Žiberna, 2007a).

11.2 Comparison and evaluation of the proposed approaches

The blockmodeling types for the generalized blockmodeling of valued networks presented in Chapter 7 are evaluated and compared in this section with indirect approaches, binary blockmodeling and among themselves. The comparison and discussion presented here is based on that presented at the end of Chapter 7. However, what is presented here also incorporates results based on the application of these approaches to empirical and artificial networks presented in Chapter 9 and results of simulations presented in Chapter 10. In addition, the indirect approaches to blockmodeling according to regular equivalence presented in Section 4.4 are discussed. At the end of this section, the properties of these approaches are further summarized.

11.2.1 Indirect approaches for regular equivalence

In Section 4.4 several indirect approaches for regular equivalence were presented. Based on existing versions of REGE (White: 2005; White and Reitz: 1985) presented in Subsection 4.4.1, several modified versions were presented. In Subsection 4.4.2, the existing REGE algorithms were modified in order to make them (more) compatible with the definition of regular equivalence used in generalized blockmodeling as discussed in Subsections 3.2.2 and 3.3.2. These versions of REGE can be classified in four groups based on whether or not they treat the two possible ties between two units jointly or separately and based on whether they demand that all unique tie values between evaluated units and a cluster or only the maximal ones are the same. The classification is presented in Table 4.1. A further distinction can be made among versions requiring that all unique tie values between the evaluated units and a cluster must be equal based on whether they produce a measure of similarity (REGGE-OC, REGGE-OCOW) or a measure of dissimilarity (REGDI, REGDI-OW). In Section 9.2 examples of networks that perfectly correspond to the equivalences that these algorithms use were constructed. The algorithms that were put in the same cells in Table 4.1 produced identical results in these idealized situations, as they use the same definitions of equivalences.

In most examples in Chapter 9 the REGE algorithms performed reasonably well (although not as good as generalized blockmodeling). Useful partitions were usually recovered, albeit often not with the desired number of clusters. There were also cases where only some clusters, usually the core ones, were well identified. Only in one of the examples (in addition to the artificially constructed one mentioned earlier) all (suitable) versions of REGE were tested.
The results obtained there show that for empirical networks algorithms that comply with the same regular definition can provide different partitions and those that comply with different definitions of regular equivalence can provide the same partitions. When four of these algorithms (each of which complies with a slightly different definition of regular equivalence) were used in the simulations in Chapter 10 all of them performed similarly (in terms of obtaining a partition as close as possible to the correct partition). Those that treat the two possible ties between two units separately performed slightly better, as was expected due to the way the networks were generated (the pair of possible ties between two units were generated as independent).

It should be noted that the REGE algorithms do not provide a partition, only a (dis)similarity matrix. Therefore, a clustering method must be used to obtain a partition. In the examples in Chapter 9 and the simulations in Chapter 10 Ward’s hierarchical clustering was mostly used.

In Subsection 4.4.3, modifications suitable for the analysis of networks where tie values are influenced by the ‘size’ (in some sense) of the units were presented. They were applied to a network of St. Marks carbon flow in Section 9.6, where they provided better partitions at least into a smaller number of clusters. However, some units were assigned to clusters into which they did not fit, yet they also did not fit into any other cluster.

11.2.2 Generalized blockmodeling versus indirect approaches

Little new can be said about the comparison of generalized blockmodeling (also for valued networks) and indirect approaches. The benefits of the optimization approach described by Doreian et al. (2005: 245) are just as relevant for valued networks as they are for binary ones. The most important advantages of generalized blockmodeling over indirect approaches are:

- the value of a criterion function as a measure of fit or the quality of a partition;
- not being limited to just structural and regular equivalence, the possibility to specify (define) equivalence by selecting the allowed block types;
- allowing the use of pre-specified blockmodels; and
- the possibility of confirmatory blockmodeling (model fitting).

However, these advantages come at a price. The generalized blockmodeling approaches are much slower then indirect approaches. This is especially true of their current implementation in the R blockmodeling package (introduced in Chapter 8). Using this implementation, only networks of a size of up to about 50 units can be analyzed. Even for such relatively small networks, it is impossible to try a more exploratory approach using many different options (combinations of numbers of clusters, allowed blocks or pre-specified models etc.). However, the implementation in Pajek (Batagelj and Mrvar, 2006) is much faster. There, networks of up to 255 units (the limit in Pajek) can be analyzed within a reasonable time. Unfortunately, Pajek currently only supports binary blockmodeling (out of the generalized blockmodeling types). Also, there is usually no assurance that the global optimum is achieved. Such an assurance can only be obtained for very small networks and numbers of clusters.
As indirect approaches are only developed for structural and regular equivalences, a more direct comparison can only be made for these two special cases.

In the case of valued networks that perfectly comply with structural equivalence, indirect approaches to structural equivalence (provided that the diagonal values in the matrix are handled correctly) provide the same partitions as homogeneity and implicit blockmodeling (according to structural equivalence).

In the case of networks that perfectly comply with max-regular equivalence, the same partitions are obtained using a ‘One Way’ version of the REGGE algorithm (REGGE-OW) and homogeneity and implicit blockmodeling according to max-regular equivalence.

When using these approaches to empirical networks in Chapter 9 the generalized blockmodeling approaches usually performed better. They found partitions with clearer blocks that allowed for a better interpretation of the results.

During the evaluation of both direct and indirect approaches to regular networks through the simulations in Chapter 10, homogeneity and implicit blockmodeling performed better than the indirect approaches. Indirect approaches, especially all versions of REGE, usually performed better than valued and binary blockmodeling when they were used without pre-specified blockmodeling. However, the simulations showed the importance of pre-specified blockmodeling. When pre-specified blockmodeling was used with binary, valued or implicit blockmodeling these approaches were among the best methods used.

11.2.3 Generalized blockmodeling types

Valued blockmodeling was mostly developed on the basis of binary blockmodeling and it is therefore not surprising that it is most similar to binary blockmodeling of all the blockmodeling types for valued networks. Binary and valued blockmodeling were already compared to some extent in Section 7.1, where valued blockmodeling was introduced. There it was also shown that, when applied to a binary network, valued blockmodeling with parameter \( m \) set to 1 actually becomes binary blockmodeling. The two remaining blockmodeling types, homogeneity and implicit blockmodeling, are very different from binary blockmodeling. Based on the results they produce, they are also very different from valued blockmodeling.

This is demonstrated in the examples in Chapter 9. Binary and valued blockmodeling often produced similar results, although valued blockmodeling often produced slightly better results. They also performed similarly in the simulations in Chapter 10. It should, however, be emphasized that almost all comparisons in Chapters 9 and 10 are made according to regular equivalence.

Binary and valued blockmodeling were only designed to distinguish among different types of blocks or connections (based on patterns of ties). In binary blockmodeling it is only important if the tie is present or not in the binarized version of the network, that is, if the tie value is larger than or equal to slicing threshold \( t \) or lower than this value. In valued blockmodeling the situation is similar, only now the difference between the tie value and either 0 or
parameter $m$ also matters. As we can see, each of these approaches has a parameter that determines which values are treated as relevant and which as irrelevant.

Ideally these parameters should be chosen before blockmodeling. When this is not the case, other approaches might be more appropriate. When this is not possible, some suggestions were made in Section 7.1.1 especially for choosing parameter $m$, however most can also be used for slicing threshold $t$, as these are closely related. These proved quite suitable in the examples in Chapter 9. However, as the suggestions are not exact, several should be tested. This is often problematic, especially for valued blockmodeling due to its slow implementation (binary blockmodeling is in addition to the blockmodeling package also implemented in Pajek (Batagelj and Mrvar, 2006), where the implementation is much faster). However, the positive side is that by choosing several different values of these parameters alternative hypothesis can be tested. Often different values of parameters can provide different partitions or blockmodels. This was best seen in the example ‘sweet’ (Section 9.5) when searching for partitions into three clusters.

In most cases the need to specify these parameters ($t$ and $m$) in advance is at least a nuisance. However if a distinction between relevant and irrelevant ties as assumed by these two blockmodeling types is supported by theory and sufficient prior knowledge to determine the appropriate values of these parameters is available, selecting these parameters can become a valuable modeling tool.

Binary and valued blockmodeling were not designed to distinguish blocks of the same type but with different values. E.g., they were not designed to distinguish between a complete block where all cell values have a value of $a$ and a complete block where all cell values have a value of $b$, where $a \neq b$. Homogeneity and impact blockmodeling (and the indirect approaches to structural and regular equivalence) can also distinguish blocks of the same type with different tie values. In the examples presented in Chapter 9 this inability did not seem to have serious consequences as the partitions that were based more on a pattern did seem appropriate. Yet it was noticed that the tie values (or values of function $f$ on the rows or columns in the case of $f$-regular blocks) in blocks induced by those partitions were not very homogeneous.

In general, binary and especially valued blockmodeling performed very well in the examples in Chapter 9, although this might partly be due to the fact that several values of parameter $m$ or the slicing threshold were used and the best solutions were then selected. However, their performance in the simulations in Chapter 10 was not as good as in examples. While they did not perform very well in simulations where networks of 11 units were generated, they performed very badly in simulations where networks of 25 units were generated. The exception was if they were used with pre-specified blockmodeling. In such cases, they were among the best methods applied.

As we discussed above, binary and valued blockmodeling were unsuitable for distinguishing blocks of the same type but with different values. While homogeneity blockmodeling is very suitable for distinguishing blocks of the same type based on tie values, it is less suitable for distinguishing blocks with similar tie values of different types. This is due to the null block
problem and problems with compatibility among certain block types. This was more thoroughly discussed in Section 7.7. In the examples presented in Chapter 9 it can be noticed that homogeneity blockmodeling often produced more homogeneous blocks (with respect to block type), however often at the cost of less clear null blocks. This negative part of this (less clear block types) is a direct consequence of the null block problem. However, in the simulations in Chapter 10 homogeneity approaches were generally the best approaches used. Only approaches with pre-specified blockmodeling performed better. In most simulations in Chapter 10, pre-specified blockmodeling was not used with homogeneity blockmodeling. Such a decision was made due to the problems of homogeneity blockmodeling described above.

It is quite surprising that the performance of valued and implicit blockmodeling differs to such an extent. As was shown in Section 7.3 where implicit blockmodeling was introduced and especially in Subsection 7.3.3, the block type inconsistencies are computed very similarly in these two blockmodeling types. The main difference is that parameter $m$ in valued blockmodeling is replaced by a block maximum (in the case of a row- or column-dominant block by a maximum of the dominant row or column) of the block analyzed. However, the effect of a relatively small difference is huge. Implicit blockmodeling is, on the other hand, also very similar to homogeneity blockmodeling as they have the same ideal blocks (if when using homogeneity blockmodeling the maximum is used as function $f$ in $f$-regular and similar blocks and if the appropriate version of row- and column-dominant blocks is selected – version 2).

Implicit blockmodeling in its original form produced results that were not very similar to those of any other approaches in the examples presented in Chapter 9. However, when used without the null block type, its results were similar to those of homogeneity blockmodeling. This is not too surprising since in that case both approaches become relatively similar. We can even consider implicit blockmodeling to be homogeneity blockmodeling with a very strange measure of variability. In the simulations in Chapter 10, homogeneity and implicit blockmodeling indeed performed relatively similarly. Surprisingly, in the simulations their performance was similar, even where implicit blockmodeling was used with a null block type.

Implicit blockmodeling can also (like homogeneity blockmodeling) distinguish blocks of the same type based on tie values (assuming that block maximums also differ) and it does not have problems with the compatibility of blocks or with the null block being a special case of other block types. However, it has its own set of problems that are mainly caused by the fact that block type inconsistencies are computed relative to the block maximum. Block type inconsistencies of implicit blockmodeling are very sensitive to changes in block maximums and this also means they are very sensitive to outliers (extremely large values). The problem is even more severe if maximum normalization is used, since in that case, the null block inconsistency of an empirical block that is not an ideal null block is actually lowered if the largest value in this block is increased. That also means the block types should be interpreted relative to the block maximum. This causes the blockmodels obtained by implicit blockmodeling (when null blocks are also used) to be inappropriate. It is simply unreasonable
to classify blocks as null based on only one extremely large value in a block. Such behavior can be seen in most examples in Chapter 9. Due to this problem, implicit blockmodeling with null blocks should not be used whenever there are extreme tie values presented in the network.

A possible solution to this problem would be to not use null blocks. However, in this case, implicit blockmodeling practically becomes a version of homogeneity blockmodeling (with a very unusual measure or variability) and suffers from the same problems, most noticeably from the null block problem. Although this solution (not to use null blocks) clearly has serious negative consequences, the results from application to examples in Chapters 9 and results of simulations in Chapter 10 showed that it leads to better partitions. The negative consequences that implicit blockmodeling is no longer able to classify blocks into null (as opposed to some other block type) is unfortunately not a serious one as the blockmodels obtained by implicit blockmodeling with null blocks were not appropriate. Regardless of that, the use of null blocks is desired if pre-specified blockmodeling can be used (that is, if sufficient theory/prior knowledge is available). In such cases, the results improve considerably, as was shown in the simulations in Chapter 10.

11.2.4 Conclusions based on comparison of approaches

Here, the results presented in the previous subsections are further summarized and are therefore not all-inclusive.

Indirect approaches were found to provide usable partitions, although usually not as good as those obtained with the direct. The ‘One Way’\textsuperscript{117} approaches performed similarly to the original ones. The versions modified to analyze networks where tie weights are influenced by the ‘size’ of the units performed better than the original ones on such networks.

The results of the comparison among generalized blockmodeling approaches are in a few respects somewhat confusing. Valued blockmodeling performed very well in the examples, but its results in the simulations (especially those where networks of 25 units were generated) were not good at all. In addition, it cannot distinguish between blocks with different values, but this does not seem to be very important as the pattern of ties seems more important ‘to the eye’. However, this might be due to the type and size of networks analyzed.

Binary blockmodeling usually performed similar or slightly worse than valued blockmodeling. This was expected, although the better performance of valued blockmodeling was overestimated. However, in the simulations binary blockmodeling sometimes even performed slightly better.

Homogeneity blockmodeling performed very well in the simulations, but in the examples this was not always the case. It produced less clear null blocks and sometimes gave too much

\textsuperscript{117} See Subsection 4.4.2 for a description of these approaches.
weight to tie values compared to the pattern of ties. This is partly due to the null block problem\textsuperscript{118}. In addition, it also suffers from compatibility problems.

Implicit blockmodeling was found to perform well only with pre-specified or without null blocks. Without using null blocks we are unable to classify blocks into null and some other block type. However, this is not a serious disadvantage as the blockmodels produced when using null blocks were inappropriate. Nevertheless, the question remains whether implicit blockmodeling without null blocks makes sense.

### 11.3 Suggestions regarding the usage of approaches to blockmodeling of valued networks

The selection of the blockmodeling approach should foremost depend on the desired characteristics of the solution, that is the type of equivalence (including generalized) we are looking for. Even if the name of the equivalences and even the ideal blocks are the same, the exact definitions of equivalences and descriptions of ideal blocks differ among the different approaches to blockmodeling, although some do share common definitions of equivalences or descriptions of ideal blocks.

The ability of some approaches to detect certain types of solutions together with some more technical aspects is discussed in the following subsection. In the event that we do not select just one approach but rather apply several of them, the selected sequence of approaches is presented in Subsection 11.3.2. The sequence presented there is designed so that the results of approaches applied in earlier phases can be used when applying the approaches applied later.

#### 11.3.1 Some suggestions on which approach to use

For the purpose of blockmodeling\textsuperscript{119} the indirect approaches have one main advantage: speed. We could also say there is another advantage of indirect approaches, which is that the number of clusters does not need to be specified in advance\textsuperscript{120}. Therefore, indirect approaches should be used either in the preliminary step of analysis or when networks are too large to be suitable for generalized blockmodeling approaches. As a preliminary step, they could be used to gain some idea about the number of clusters, possible values of parameter $m$ for valued blockmodeling or threshold $t$ for binary blockmodeling etc. They can also be used to find a starting partition that can later be optimized by using some generalized blockmodeling type (using a local search). However, this is only meaningful if we want to partition the network in terms of structural or regular equivalence.

\textsuperscript{118} The null block problem by the null block is a special case of all other block types, especially (also in terms of the way inconsistencies are computed) of the complete block.

\textsuperscript{119} Another advantage of the indirect approaches is that they produce a (dis)similarity matrix which can also be used for purposes other than blockmodeling, e.g. multidimensional scaling, fuzzy clustering etc.

\textsuperscript{120} However, this is mainly only relevant due to the first advantage. If indirect approaches were faster, the number of clusters could be just as easily determined by running them with a different number of clusters.
As mentioned before, generalized blockmodeling approaches are only feasible for smaller networks. The binary blockmodeling implemented in Pajek (Batagelj and Mrvar, 2006) can now be used on networks with up to 255 units. Other generalized blockmodeling types implemented in the R ‘blockmodeling’ package (Žiberna, 2006) can only be used on networks with about 50 units. It is hoped that with better implementations (especially for valued, homogeneity and implicit blockmodeling) and ever faster computers larger networks can be analyzed.

When these approaches are feasible, binary blockmodeling is best suited for situations where either an ordinal (or binary) scale is used for measuring tie values when it is only relevant if the tie value is lower than or larger than (or equal to) some pre-specified value. The researcher should at least have some idea what this pre-specified value should be.

Other generalized blockmodeling approaches can only be used on networks measured on at least an interval scale. Valued blockmodeling is most suitable when networks are measured on a scale with two well-defined extremes (e.g., 0 and the maximum of the scale) and we are only interested in whether the tie values are closer to 0 or the maximum. It can also be used on other networks when we are primarily interested in whether the values are closer to 0 or to some pre-specified value (parameter $m$), but we do not care whether they are larger than or equal to that pre-specified value ($m$). Like should be done when using binary blockmodeling, when using valued blockmodeling the researcher should also have at least some idea of what this pre-specified value should be. If this is not the case, either homogeneity or implicit blockmodeling should be used or some other approach (indirect blockmodeling, homogeneity, or implicit blockmodeling) should be used to find an approximate partition. This partition can be used to find suitable values for these pre-specified values by checking what kind of blockmodels we would obtain from this partition using different values for pre-specified values. There are also other ways of finding these values without the need for a partition, however the estimates obtained using a partition are usually better.

Valued and binary blockmodeling can only be used to classify empirical blocks into block types, but cannot be used to distinguish blocks of the same type based on tie values (see the previous subsection). For that purpose, homogeneity and implicit blockmodeling should be employed. Homogeneity blockmodeling (especially sum of squares blockmodeling) is usually most suitable for distinguishing blocks of the same type based on tie values. However, due to the problems described in the previous subsection homogeneity blockmodeling is less suitable for distinguishing blocks based on the block types.

Implicit blockmodeling can, on the other hand, distinguish blocks based on both tie values and block types, however the classification it produces is very dependent on the block maximums and should be interpreted relative to the block maximums. Due to this problem, the usefulness of the returned blockmodel and partition is questionable.

The use of pre-specified blockmodeling is advised whenever sufficient prior knowledge or hypotheses are available with any generalized blockmodeling type, although it is especially desired when used with binary and valued blockmodeling. When used without pre-specified
blockmodels (especially if we allow a larger number of allowed block types or if the allowed block types include less restrictive block types such as regular, row- and column-regular, row and column-dominant etc.), binary blockmodeling in particular can produce a large number of partitions with the same (minimal) total inconsistency.

11.3.2 Suggested sequence of methods when several are used

Often several approaches are applied to one network. This is either done for the purpose of complementing the results of these approaches or because we cannot determine which approach is most suitable solely based on theory and prior knowledge about the network.

The sequence of approaches suggested here is only optional. It is designed for the purpose of first using the faster approaches and those that need less prior knowledge. Following this strategy, the indirect approaches should be used first. Their results can be used in several ways. First, they give an indication of whether structural or regular equivalences are suitable theories for partitioning the network. If they are, based on their results we can assess the suitable number of clusters and they also provide us with a starting partition. This partition can be used as a starting partition (important especially for larger networks) for local optimization with generalized blockmodeling approaches. Looking at the matrix ordered (and split) based on this partition, we can also get an idea of which block types should be used with generalized blockmodeling. If we also intend to use binary or and valued blockmodeling, this partition can also be used to determine possible values of threshold $t$ used for binarization in binary blockmodeling and/or parameter $m$ used in valued blockmodeling.

After the indirect approaches, homogeneity and/or implicit blockmodeling should be used as they do not have an extra parameter such as threshold $t$ (binary blockmodeling) or parameter $m$ (valued blockmodeling). If we are dealing with a larger network, we can start local optimization (preferably in addition to several random starting partitions) for the partition(s) obtained using indirect approaches or any other methods. The results obtained using these two blockmodeling types can be employed in similar ways to those obtained with indirect approaches for use with binary and/or valued blockmodeling (and for use with implicit blockmodeling if homogeneity blockmodeling was employed first, or vice versa). However, it is expected that the results obtained using these generalized blockmodeling approaches will provide better grounds for other generalized blockmodeling approaches than those obtained using indirect approaches since they have more in common.

11.4 Further research

During the work on this thesis, several new questions arose or existing open questions were left unanswered in order to limit the extent of this thesis. Some of them were already pointed out, usually those that deal directly with the disadvantages and other properties of the proposed approaches. Others are mentioned here for the first time. Several alternatives and suggestions for improvement were presented in Section 7.6. Only the most important of these and of those mentioned elsewhere are presented here.
11.4.1 Improving/testing homogeneity blockmodeling

One problem that was frequently pointed out in connection with homogeneity is the null block problem. Some solutions were already suggested in Section 7.4. However, much research is still needed on this subject. At a minimum, the suggestions presented in Section 7.4 should be implemented and tested. This is a very important problem for at least two reasons. Without null blocks, the distinction between null and other block types is not possible within homogeneity blockmodeling and this is the most important distinction between blocks. Also, without null blocks, pre-specified blockmodeling can hardly be used. As was shown in the simulations in Chapter 10, the use of pre-specified blockmodeling considerably improves the results of blockmodeling.

The other problem that also occurs within homogeneity blockmodeling is the problem of the compatibility of $f$-regular and similar block types with other block types. The question that should be explored here is whether $f$-regular and similar block types are compatible with other block types. If the answer is in general ‘no’, whether they are compatible for some function $f$ should be explored. For now, it was determined that they are compatible with other block types at least if function $f$ is the mean. If they are not compatible (or for functions $f$ for which they are not compatible), it should be explored if these block types could be modified so that they would become compatible. This is quite a problem since the central idea of generalized blockmodeling is that blocks of different types can be used together.

These two problems considerably limit the usefulness of homogeneity blockmodeling, an approach that was found to be the best approach in the simulations in Chapter 10 in the situations where these two problems did not seriously affect the results (where pre-specified blockmodeling was not used and networks were generated based on regular equivalence). As a result, these are some of the most important areas to be studied.

11.4.2 Improving implicit blockmodeling

Implicit blockmodeling was found to have several very desirable properties. However, some negative ones were also identified which usually prevailed when put into practical use when used with null blocks (and without pre-specified blockmodeling). Using implicit blockmodeling without null blocks dramatically improved the partitions produced by this approach. However, it also eliminated several of the desirable properties.

A solution that might also improve the blockmodels and partitions produced by implicit blockmodeling was presented (as an untested idea) in Subsection 7.6.6. The key idea is to replace the maximum with something less sensitive than the extreme value and to set a lower limit on the value from which the deviations are computed in blocks other than null (currently maximum). These two modifications are sufficient if maximum normalization is not used. As mentioned, this idea is described in more detail in Subsection 7.6.6. It is hoped that by such

---

121 row-$f$-regular and column-$f$-regular block types
modification implicit blockmodeling can also in practice show all of the desirable properties and avoid the negative ones. Then implicit blockmodeling could become the best approach to the generalized blockmodeling of valued networks (or one of the best if the problems of homogeneity blockmodeling presented in the previous subsection are solved).

11.4.3 Extending generalized blockmodeling to ordinal valued data

Generalized blockmodeling has been developed for binary and valued networks, yet many social networks are measured on an ordinal level. The problem with ordinal data is how to compute the difference between two values. A subtraction is not allowed on ordinal variables as the difference between two subsequent categories is not necessarily the same. On the other hand, to assign the same difference to all unequal values means losing a lot of information. This problem is usually resolved by some transformation of the data.

The simplest option often used when dealing with ordinal values is to convert them to ranks and then to use the methods or approaches for interval valued data. Such an approach is, for example, taken with Spearman’s rank correlation coefficient. With such modifications, ordinal valued networks can be analyzed using homogeneity blockmodeling. The problem with this approach is that it also changes zeros to some other numbers, which is problematic for valued and implicit blockmodeling, although this might be corrected by deducting the minimal rank from all values.

Other transformations could also be employed, such as those based on the assumption that ordinal variables are measurements of an underlying interval (continuous) variable with a known distribution (actually, the transformation into ranks can also be seen as a transformation of this kind, when the underlying distribution is a uniform distribution). If such a transformation is appropriate, the transformed networks can then be (as before) analyzed using homogeneity blockmodeling. If the value 0 retains its special meaning or if some additional transformations are used to ensure that, then valued and implicit blockmodeling can also be utilized.

When analyzing data measured on ordinal scale close to the interval scale, the methods created for the interval data are also often used without a modification of the prior transformation. In such cases, the methods presented in this thesis could be employed.

11.4.4 Better implementation of generalized blockmodeling of valued networks

Generalized blockmodeling of valued networks is currently only implemented in a blockmodeling package (Žiberna, 2007a) for statistical program and environment R (R Development Core Team 2006). However, this implementation (an exception is the implementation for sum of squares blockmodeling for structural equivalence) is extremely slow and therefore only useful for testing and application on very small networks.

At the very least, the algorithms used should be implemented in some lower level programming language (in order to speed up the process). Additional research on ways of optimizing the code would be useful here.
Only after a faster implementation is available can the approaches presented in this thesis be considered truly useful to the scientific community.

11.4.5 New block types
In this dissertation, the block types for approaches to the generalized blockmodeling of valued networks were created based on block types for generalized blockmodeling of binary networks. Although these block types are appropriate and useful for the generalized blockmodeling of valued networks, some others also might be. For specific blockmodeling problems new blocks could be needed. Although the construction of new block types would be a useful line of research, it is best done when facing blockmodeling problems that cannot be adequately solved using existing block types, armed with theory and expectations about appropriate block structures.

11.4.6 Using generalized blockmodeling on multi-relational networks
Most real life networks are multi-relational networks. While the theory of generalized blockmodeling presented in this thesis is mainly presented for one-relational networks, extensions to multi-relational networks are possible. The simplest extension is to sum up the block inconsistencies for all relations. We can allow different blockmodels in different relations (while demanding the same partitions of units) or demand the same blockmodels in all (or just some) relations. This is relatively simple to do if all relations are binary or at least measured on the same scale and we are using the same blockmodeling type for all relations. If inconsistencies in some networks are more important than in others, we simply compute the weighted sum instead of the ordinary one using appropriate weights. However, Doreian et al. (2006: 357) suggest caution when using several relations together, especially if they are quite different. If the relations are quite different, they see no point in using them together in blockmodeling and thus treating them all essentially the same. They also point out that ‘Stacking markedly different relations in the indirect approach seems a guarantee for generating confusion’ (Doreian et al., 2006: 357).

Further research is needed to determine what to do when relations are not measured on the same scale or when different blockmodeling types are used. The second problem is further discussed in the next subsection. If relations are measured on different scales but are all (at least interval valued), one solution is to normalize them while the other would be to appropriately weight the inconsistencies obtained for the different relations. Both options call for further research.

11.4.7 Using several generalized blockmodeling types together on the same (one-relational) network or on multi-relational networks
There may be instances where we would like to use several blockmodeling types together. For one-relation networks, the reason for using several blockmodeling types (on the same relations) would be to try to correct some of the deficiencies of certain blockmodeling types. For example, using binary blockmodeling with homogeneity blockmodeling was suggested as
a possible solution to the null block problem. Using binary blockmodeling with implicit blockmodeling was also suggested as a possible solution to the problem of implicit blockmodeling that leads to any block with one extremely large value being declared null.

The use of binary blockmodeling with practically any other blockmodeling type (for valued networks) is also suggested when an additional error is desired if some tie value is not just lower or higher than it is in the ideal block but a tie is missing (or present where it should be missing).

With multi-relational networks there is even a greater need to use several blockmodeling types together. The simplest reason might be that relations are measured on scales of a different measurement type (binary versus interval) or there might be other reasons why different blockmodeling types might be appropriate for different relations. In addition, even for a single relation the use of several blockmodeling types might be desired for the reasons described above.

The main problem (in addition to those reported in the previous section) in all these scenarios is how to combine inconsistencies based on different blockmodeling types (and possibly different relations). If we want to sum up the block inconsistencies based on different blockmodeling types, there is a need to normalize these inconsistencies or to employ appropriate weights (or both). However, finding appropriate weights or ways of normalizing the block inconsistencies needs further research, although some suggestions especially for using valued and implicit blockmodeling with binary blockmodeling were given in Subsection 7.6.1.

11.4.8 Multi-mode and multi-way blockmodeling

The mode of a network is the number of distinct sets of units in the network. The ties are only present between sets. All the networks analyzed in the dissertation are one-mode networks since they have only one set of units (only people, only organizations, only species etc.). Two-mode networks have two sets of units (e.g. people and organizations, authors and papers etc.), three-mode networks have three sets of units etc. The generalized blockmodeling approaches presented in the dissertation can also be used for blockmodeling two-mode networks. This is also implemented in the blockmodeling package (Žiberna, 2007a). However, the generalized blockmodeling of valued networks has not yet been sufficiently tested on two-mode networks. Indirect blockmodeling according to structural equivalence for two-mode networks is essentially a classical cluster analysis for a standard data structure (two-mode matrices – cases by variables). REGE as the family of algorithms for indirect blockmodeling according to regular equivalence can also be easily adapted to two-mode networks, although it can also be used in its one-mode version (as can indirect blockmodeling according to structural equivalence) provided that the two-mode matrices representing the network are transformed into one-mode networks (with units of both sets) and a constraint that units from different sets (original modes) cannot be clustered together is applied.
In multi-way networks (matrices), the tie is defined between several units. The number of units between which the tie is defined equals the number of ways of the network (matrix). All approaches presented here are only applicable to two-way networks (matrices). The theoretical foundations for the blockmodeling of multi-way, multi-mode matrices were laid by Borgatti and Everett (1992b). They also proposed definitions of structural and regular blockmodels for multi-way matrices. However, their definition of structural blockmodels suffers (where some ways refer to the same objects) from the same shortcomings as some early definitions of structural equivalence (Lorain and White, 1971: 63; Burt, 1976: 96) for one-more two-way matrices (networks). Batagelj et al. (2007) recently presented a formal definition of structural equivalence for three-way (one-, two- and three-mode) matrices (networks) without these shortcomings and an indirect approach to the blockmodeling of three-way (one-, two- and three-mode) matrices according to structural equivalence. The approaches to the generalized blockmodeling of valued networks presented in his thesis can hopefully also be extended to three-way networks using the ideas of Borgatti and Everett (1992b) and Batagelj et al. (2007).

11.4.9 Missing data and network boundary problem
As also pointed out by Doreian (2006: 126,127), generalized blockmodeling’s vulnerability to missing data and an inappropriate specification of the network boundary still needs to be assessed. Generalized blockmodeling assumes that the complete network is observed and that the boundary of the network is specified correctly.

Missing data can occur in two forms, either as missing information for only certain tie variables (we do not know if a tie from unit \( a \) to unit \( b \) exists and, if it exists, what is its value in the case of a values network). The second form is that all data concerning a specific unit is missing. In the case of this second form, the unit for which the data are missing is dropped from the analysis. As mentioned, the effects of omitting a unit still need to be studied. It is, however, clear that the effects differ significantly depending on the type of equivalence used or on the type of blocks in which the ties of such a unit are present. E.g., the omission of a unit which only has ties in null and complete blocks (especially if the network perfectly corresponds to the selected equivalence) and does not constitute its own cluster does not affect the partition of the remaining units. On the other hand, units forming a cycle in a directed network (and which are disconnected from any other potential units) are regularly equivalent to each other. However, if any one of them is removed each of the remaining units constitutes its own regular equivalence class. We can imagine a similarly devastating effect if we remove a unit that is the dominant unit in a row- or column-dominant block. These are, of course, just a few examples where only a perfect fit to equivalences was considered. However, they show that missing data can have serious consequences for blockmodeling and that there is a need to explore the effect of missing units on the results of generalized blockmodeling. In addition, different imputation techniques for dealing with missing tie variables should be considered and their effect on the solutions of generalized blockmodeling explored. Effects of both versions of missing data could be studied, as suggested by Doreian.
by either declaring some data as missing in the real complete data or through simulations.

A closely-related problem to missing units is the network boundary problem. This problem occurs when we are unsure which units should be included in the network. Which units should be included in the network is usually clear when analyzing the formal organizations (e.g. pupils in a class at a school, members of a certain organizations, scientists employed at a certain institution etc.), however this could be very problematic especially in some other less formal settings (e.g. scientists working in a specific scientific field). Laumann et al. (1983) presented a categorization of different approaches used to specify the boundary of the network and emphasized the need for more explicit attention to boundary specification. Doreian and Woodard (1994) suggested an empirical procedure for specifying the boundary of a network. However, Doreian (2006: 126) states that often the specified boundary still omits other relevant units. Doreian and Woodard (1994: 291) also noted that the correct specification of the boundary problem is particularly important when performing positional analysis, of which blockmodeling forms part of.

If some relevant units are excluded, this problem is similar to the missing unit problem. The situations differ slightly, mainly in terms of what causes the units to be excluded from the network. As a result, we can assume the most central or well-connected units will not be excluded due to the boundary problem. The other possible effect of an inappropriate boundary specification is that certain units are included in the network which should not be. In the case of structural equivalence, one additional unit can in principle (an extreme option) in the case of binary directed networks split each structural equivalence class into four classes and into even more in the case of valued networks. Similarly as removing one unit from (an isolated) cycle in a directed network makes the units previously occupying the cycle no longer regularly equivalent, adding a unit that has a unidirectional tie to only one of these units has the same effect.

Clearly, missing data and inappropriate boundary specification can cause considerable problems for blockmodeling. Here I note some examples of the effect of adding or removing units can have on partitions that perfectly comply with a certain definition of equivalence (e.g. structural, regular, generalized). However, in practical applications we usually do not demand that a partition perfectly comply with a certain definition of equivalence, but rather search for a partition that fits reasonably well with the selected definition of equivalence. The effect of missing data and inappropriate boundary specification should also be studied in more detail in such contexts, preferably by using real and (realistically) simulated networks and then imposing missing units (or ties) and a wrong boundary specification on them. Of course, when using real data the study of the boundary problem is more problematic as it is hard to determine (as noted above) what the correct specification of the boundary is in the first place. Nevertheless, the effect of using different boundary specifications could be studied.
12 REFERENCES


Rape, Tamara (2005): Socialne delavke v delavnih organizacijah s poudarkom na njihovem povezovanju in sodelovanju: Diplomska naloga. FSD, Ljubljana.


The Edinburgh Associative Thesaurus (EAT) (2004): Available at http://www.eat.rl.ac.uk/ (February 24, 2004).


13 AUTHOR INDEX

Arabie, Phipps, 8, 22, 25, 26, 41, 147, 167, 173, 257

Baird, Daniel, 149, 163, 281

Batagelj, Vladimir, xii, 1, 2, 3, 6, 8, 11, 13, 21, 22, 27, 28, 34, 42, 44, 46, 47, 49, 50, 51, 54, 58, 68, 82, 84, 85, 94, 98, 137, 164, 207, 210, 212, 215, 217, 221, 227, 232, 243, 244, 246, 248, 249, 250, 252, 254, 255, 258, 259, 260, 262, 263, 268, 271, 273

Berkowitz, Stephen D., 6

Boorman, Scott A., 8, 22, 25, 26, 41, 257

Borgatti, Stephen P., vii, 5, 9, 11, 12, 13, 14, 15, 16, 17, 19, 20, 22, 23, 28, 29, 30, 31, 32, 33, 34, 39, 42, 43, 45, 54, 68, 149, 150, 153, 163, 227, 255, 256, 258, 273, 280, 281, 282

Boyd, John, 12, 14, 23

Brandes, Ulrik, 25, 39, 40, 41, 258

Breiger, Ronald L., 8, 15, 16, 22, 25, 26, 41, 42, 44, 45, 257, 258

Bren, Matevž, 28

Burt, Ronald S., 8, 12, 25, 26, 28, 227, 257

Carrington, Peter J., 7

Christian, Robert R., 149, 163, 281

Degene, Alain, 7

Doreian, Patrick, xii, 1, 2, 8, 11, 13, 21, 22, 23, 34, 36, 42, 44, 46, 47, 49, 50, 51, 54, 57, 58, 68, 91, 94, 209, 210, 215, 227, 228, 243, 244, 248, 249, 250, 254, 255, 258, 259, 260, 262, 263, 273

Everett, Martin G., vii, 5, 9, 11, 12, 13, 14, 15, 16, 17, 19, 20, 22, 23, 28, 29, 30, 31, 32, 33, 34, 39, 42, 43, 45, 54, 68, 149, 150, 153, 163, 227, 255, 256, 258, 273, 280, 281, 282

Faust, Katherine, 1, 4, 6, 7, 8, 9, 10, 15, 26, 28, 30, 32, 243, 248, 254

Ferligoj, Anuška, xii, 1, 2, 8, 11, 13, 21, 22, 27, 34, 42, 43, 44, 46, 47, 49, 50, 51, 54, 58, 68, 94, 178, 210, 215, 227, 243, 244, 248, 249, 250, 254, 255, 258, 259, 260, 262, 263, 273

Forse, Michel, 7

Freeman, Linton C., 16, 22, 28, 29, 30, 43, 45, 258

Gelatt, C.D., 43

Glover, Fred, 43

Goldberg, David E., 43

Hlebec, Valentina, 122

Homans, George C., 22, 43

Hubert, Lawrence, 147, 167, 173

Johnson, Jeffrey C., vii, 149, 150, 153, 163, 280, 281, 282

Kirkpatrick, Scott, 43

Knoke, David, 9

Kuklinski, James H., 9

Lerner, Jürgen, 25, 39, 40, 41, 258

Lorrain, Francoise P., 8, 10, 11, 16, 255

Luczkovich, Joseph J., vii, 149, 150, 153, 163, 280, 281, 282

Mandel, Michael J., 5, 9, 10, 13, 14, 23, 256

Mohr, John W., 15, 16, 42, 44, 45, 258

Moreno, Jacob L., 4

Mrvar, Andrej, 6, 27, 28, 57, 91, 98, 137, 164, 207, 212, 215, 217, 221, 246, 252, 271

Nordlund, Carl, vii, 36, 39, 149, 150, 153, 157, 159, 163, 258, 282

Nowicki, Krzysztof, 22, 258

R Development Core Team, 214, 232
Rape, Tamara, 112
Reitz, Karl P., 5, 11, 12, 13, 30, 31, 32, 255
Sailer, Lee Douglas, 26
Scott, John, 7
Snijders, Tom A. B., 22, 258
Stadler, Peter F., 14
Tinhofer, Gottfried, 14
Vecchi, Mario P., 43
Ward, Joe H., 27, 135
Wasserman, Stanley, 1, 4, 6, 7, 8, 9, 10, 15, 26, 243, 248, 254
Wellman, Barry, 6, 9
White, Douglas R., 5, 11, 12, 13, 15, 17, 29, 30, 31, 32, 33, 122, 210, 244, 249, 255, 256, 258
White, Harrison C., 8, 10, 11, 16, 255
Winship, Christopher, 5, 9, 10, 13, 14, 23, 256
Wu, Lawrence L., 23
Zaveršnik, Matjaž, 134
Žiberna, Aleš, 2, 3, 97, 98, 170, 206, 211, 214, 221, 224, 226, 246, 251, 255, 270
14 SUBJECT INDEX

actor, 4, 6, 8, 9, 10, 150
Adjusted Rand Index, 147, 148, 167, 170, 173, 174, 178, 179, 188, 190, 192, 195, 200, 201, 279, 291
algorithm
clustering, 29, 131
CONCOR, 22, 25, 26, 41, 257
hierarchical, 29
analysis
cluster, 25, 26, 226, 280
network, 1, 2, 4, 6, 7, 8, 10, 11, 13, 23, 230, 243, 244
positional, 228, 231
approach
direct, 1, 2, 3, 15, 22, 42, 43, 45, 46, 51, 109, 124, 164, 170, 180, 190, 195, 199, 204, 207, 208, 210, 213, 243, 244, 247
indirect, 1, 2, 3, 22, 25, 26, 41, 42, 54, 88, 93, 94, 96, 100, 114, 123, 124, 135, 136, 165, 180, 190, 195, 196, 208, 210, 212, 214, 215, 216, 217, 219, 220, 222, 225, 227, 243, 244, 246, 247
optimizational, 46
arc, 5
averaging rule, 3, 82, 83, 84, 158, 159, 160, 162, 163, 166, 213, 247
block, 8, 10, 13, 48
allowed, 2, 11, 20, 46, 48, 55, 57, 74, 76, 78, 85, 87, 88, 94, 98, 101, 148, 165, 168, 169, 170, 204, 208, 215, 222, 244, 275, 279
column-dominant, 51, 54, 55, 58-61, 64, 65, 68, 71-73, 84, 87, 95, 99, 167, 204, 209, 218, 222, 227, 277
column-f-regular, 55, 58-60, 66-78, 86, 87, 92, 93, 96, 98, 99, 223
column-functional, 50, 51, 54-56, 59, 61, 62, 65, 68, 69, 71, 73, 74, 84, 87, 99, 209
column-regular, 36, 50, 51, 52, 56, 84, 85, 99, 150, 163, 222
empirical, 46-49, 52, 54, 55, 57, 77-79, 81, 82, 84, 85, 96, 102, 116, 133, 158, 164, 200, 211, 218, 221, 245, 275
empirical, 48
f-regular, 55, 60, 61, 67, 86, 88, 92, 93, 127, 160, 169, 170, 180, 217
ideal, 15, 21, 47-50, 52, 53, 55-57, 59, 61-67, 69, 78, 79, 82, 84, 88, 95, 96, 102, 109, 123, 128, 133, 158, 164, 200, 210, 212, 218, 220, 226, 244, 246
inconsistency, 17, 47-50, 54, 57, 63-68, 72-74, 78, 79, 81, 82, 85, 86, 89-92, 96, 97, 103-108, 118, 146, 165, 218, 225, 226, 275, 276
279
off-diagonal, 26, 70, 277
row-dominant, 50-52, 55, 56, 58, 59, 61-65, 69, 70, 73, 84, 87, 99, 204, 277
row-regular, 36, 50-51, 55, 56, 59, 61-65, 69, 70, 73, 84, 87, 99, 204, 277
row-regular, 36, 50, 51, 55, 84, 99, 150, 163
blockmodel, 2, 8, 10, 48, 55, 57, 60, 68, 74, 88, 92, 93, 101-103, 105, 106, 115, 117, 118, 124, 126, 127, 133, 137, 140, 149, 150, 157, 158, 161, 163, 165, 167, 168, 170, 171, 175-177, 179, 180, 183, 191, 192, 195, 197, 204, 209, 211, 212, 217-223, 225, 229, 233, 234, 244, 245, 246
indirect, 2, 40, 41, 167, 221, 226
structural, 42, 227
blockmodeling, 8, 9, 11, 25, 210, 243
package, 2, 3, 94, 97-99, 206, 211, 214, 215, 217, 224, 226, 246
pre-specified, 9, 44, 67, 74, 76, 80, 81, 93, 103, 104, 106-108, 168, 170, 179, 180, 188, 190, 192, 195, 197-201, 204, 209, 213, 216-219, 221, 223, 247
sum of squares, 46, 67, 68, 78, 86-88, 92, 96, 107, 117, 118, 120, 134, 149, 165, 166, 169, 179, 180, 196, 200, 201, 207, 208, 213, 221, 224, 247, 278
cluster analysis, 25, 26, 226, 280
coloration, 14, 20, 40
ecological, 14
exact, 14, 19, 20, 39, 40, 231
regular, 14
compatible, 25, 26, 29, 41, 67, 75, 78, 79, 92, 96, 159, 210, 214, 223, 244
CONCOR, 22, 25, 26, 41, 257
contingency table, 16, 42, 44, 45
criterion function, 1, 2, 21, 22, 42-44, 47-53, 55, 57, 61, 88-91, 93, 94, 98, 99, 101-106, 108, 112, 133, 139, 164, 170, 178, 196, 200, 210, 215, 243, 244
binary, 28
valued, 224
dendrogram, 114, 115, 118, 123, 135, 153
density, 50, 51, 56, 57, 59, 60, 84, 99, 138, 169, 171
dissimilarity, 1, 22, 25, 26, 28, 29, 34, 39, 40, 214, 243
corrected Euclidean-like, 29, 114, 123, 168, 208
distance, 27, 33, 40, 94
Euclidean, 27, 28
Manhattan, 29
truncated Euclidean, 28, 29
distribution, 19, 60, 74, 101, 134, 142, 143, 144, 159, 168, 169, 171, 172, 175, 188, 206, 211, 224, 245
beta, 171, 172, 187, 188, 205, 206, 208
of tie values, 134, 188
edge, 5, 10, 12
eigenvalue, 40
automorphic, 11, 13, 14, 20
f-regular, 18-20, 22, 23, 55, 120, 133, 145, 146, 159, 168-170, 210, 244, 279
generalized, 1, 2, 8, 9, 11, 20, 21, 23, 44, 52, 55, 204, 205, 209, 214, 243, 244,
other, 11, 13, 14, 20
directed, 5, 39, 40
undirected, 10, 39, 40, 41
group, 22, 38, 39, 40, 43, 55, 149, 177-180, 190, 192, 195, 214, 231, 232
hierarchical clustering method, 26, 27, 131, 153
average, 27, 28
complete linkage, 28, 30, 131
Lance-Williams recurrence formula, 27
maximum, 27
single linkage, 27, 28, 29, 131
image, 43, 75, 76, 82, 98, 104, 116, 117, 120, 121, 128, 150, 157-160, 162, 163, 166, 172, 213, 247, 275, 280
image matrix, 43, 98
block, 17, 47-50, 54, 57, 63-68, 72-74, 78, 79, 81, 82, 85, 86, 89-92, 96, 97, 103-108, 118, 146, 165, 218, 225, 226, 275, 276
block type, 48-52, 57, 58, 61, 67-70, 72-75, 78-80, 82, 85, 89-92, 94, 96, 104, 108, 118, 218, 275, 276
total, 48, 75, 104-108, 139, 142, 144-147, 170, 196, 200, 222
loop, 12, 16, 28, 29
binary, 31, 36
dissimilarity, 25, 28, 34
image, 43, 98
similarity, 29, 31, 32, 131, 215, 220
valued, 5, 52, 116
measure
dissimilarity, 26, 40
similarity, 28
model, 6, 8, 9, 17, 22, 42, 44, 46, 60, 74, 94, 103-106, 129, 133, 136-138, 141, 142, 215, 230, 253, 259, 272
core-periphery, 109
multidimensional scaling, 25, 26, 220
network, 1, 2, 4, 5, 6, 7, 10, 210, 244
artificial, 39, 94, 101, 103, 109, 214
binary, 3, 8, 11, 14, 15, 18, 19, 21, 23, 30, 34, 39, 46, 47, 49, 50, 54, 55, 57, 58, 82, 91, 93, 94, 109, 114, 130, 138, 168, 170, 171, 174, 175, 190, 206, 216, 225
directed, 17, 39, 41, 227, 228
multi-relational, 19, 31, 225, 226
normalized, 93, 150, 161, 166, 213, 247
one-mode, 6, 16, 44, 100, 101, 226
one-relational, 31, 225
signed, 1, 2, 91, 243, 244
social, 1, 4, 6, 7, 9, 10, 11, 13, 23, 101, 224, 230, 231, 232, 234, 243
three-mode, 226
three-way, 227
two-mode, 6, 44, 100, 101, 210, 226, 231, 244
undirected, 101, 111, 112
valued, 1-3, 5, 7-9, 11, 14-24, 28, 30, 31, 34, 35, 39, 42, 44-48, 51, 52, 54, 55, 57, 68, 76, 77, 82, 85, 89, 91, 93, 94, 114, 115, 122, 130, 150, 153, 163, 164, 166, 167, 171, 175, 179, 196, 206, 208, 210, 213-216, 224-228, 232, 243-245, 247
normalization, 36, 70, 72, 74, 75, 92, 93, 97, 106, 107, 183, 286
block size, 70, 73-76, 146, 175, 183-188, 275, 289
maximum, 72, 74, 75, 90, 92, 97, 146, 175, 183, 211, 218, 223, 245
optimization, 80, 81, 94, 102, 170, 174, 196, 199-201, 207, 209, 215, 232
local, 102, 170, 197, 199-202, 206, 207, 222
Pajek, 4, 6, 7, 27, 28, 98, 100, 137, 164, 206, 212, 215, 217, 221, 229, 230, 232, 246, 259
parameter
m, 10, 19, 20, 33, 34, 47, 54-57, 59, 60, 68, 71, 72, 75, 77, 85, 86, 90-95, 102-106, 109, 117, 118, 125-127, 129, 133, 140, 142, 144, 146, 148, 149, 159-161, 164-166, 168-170, 197, 212, 213, 216-218, 220-222, 246, 247, 250, 252, 253, 262-264, 267, 269-271, 273, 275, 279
maximal regular, 109, 114, 180, 196
regular, 109, 110, 118, 144, 166, 208, 212, 213, 246, 247
position, 9, 10, 12, 13, 23, 39, 42, 46, 51, 58, 63-66, 70, 73, 77, 84, 87, 89, 90, 171
pre-specified, 9, 22, 44, 46, 50, 51, 67, 74, 76, 80, 81, 93, 94, 103-108, 168, 170, 179, 180, 188, 190, 192, 195-201, 204, 208, 209, 213, 215-221, 223, 247
problem
blockmodeling, 225
null block, 3, 77, 90, 97, 165, 212, 218-220, 223, 226, 246
procedure
clustering, 40
15 ABSTRACTS

15.1 English abstract

The aim of this dissertation is to develop approaches to the generalized blockmodeling of valued networks, as up till now, the generalized blockmodeling approach has only been developed for binary and signed networks by Doreian et al. (2005), although Batagelj and Ferligoj (2000: 12-13) have also presented some initial ideas extensions to valued networks.

In addition, existing approaches that could be used for the blockmodeling of valued networks are reviewed and evaluated. For one such approach, an indirect approach for regular equivalence, some modifications are also suggested. This is also done with the aim to find the best methods or approaches for blockmodeling of valued networks that can find solutions with different characteristics on networks of different types.

15.1.1 Introduction to generalized blockmodeling

Generalized blockmodeling or more generally blockmodeling is a part of (social) network analysis. Simply put, a network is a set of units with one or more relations defined on them. Different kinds of networks exist, however only complete (and not ego-centered) valued networks are discussed here. Wasserman and Faust (1994: 3-4) stated that network analysis is based on the assumption of the importance of relationships (relations) among the interacting units. Blockmodeling is a method for partitioning units in the networks and determining the pattern of relations among clusters. Blockmodeling seeks clusters of equivalent units based on some notion of equivalence. As noted by Doreian (1988a: 243) ‘Equivalence has become a foundational concept in social network analysis’. The two most widely used equivalences are structural and regular equivalence.

For the purpose of this dissertation, one of the most important divisions of blockmodeling approaches is on indirect and direct approaches. The indirect approaches first compute some measure of similarity or dissimilarity among units based on a selected measure of equivalence and then use one of the classical clustering techniques to uncover clusters of unit, while the direct approaches directly search for a partition that best fits the selected equivalence as is measured by a selected criterion function (Batagelj et al., 1992b: 66).

Generalized blockmodeling is based on a direct approach. When compared to other direct approaches, its main strength is its adaptability. It can be used to perform blockmodeling according to different types of equivalences, including generalized equivalence. Generalized equivalence is not a specific type of equivalence but more of a concept for building ‘custom-made’ equivalences. It is defined by specifying allowed types of connections between and within clusters. However up till now, generalized blockmodeling was developed only for binary and signed networks.

In this dissertation new approaches to generalized blockmodeling of valued networks are developed. However, as they can still be regarded as approaches to generalized
blockmodeling as presented by Doreian et al. (2005), the same type of criterion function can be used. The most important differences between the approaches to generalized blockmodeling proposed by Doreian et al. (2005) and those developed here is that approaches proposed by Doreian et al. (2005) can only be applied to binary and signed networks, while those presented in this dissertation can be applied to valued\textsuperscript{122} networks. To achieve that, new block types appropriate for valued networks are introduced. The common characteristics of all approaches to generalized blockmodeling (including those developed here) are, in addition to the common basic criterion function, also their ability to specify the desired solution either by a type of equivalence (which is then translated into allowed block types) or by generalized equivalence. Generalized equivalence is defined directly by the allowed block types or even more precisely by the desired blockmodel.

15.1.2 A short overview

To put these approaches into prospective, networks, network analysis, blockmodeling and equivalences are first reviewed in Chapters 2 and 3. The most important contribution of these two chapters is the discussion of equivalences for valued networks, where a new equivalence is proposed. The proposed equivalence is called $f$-regular equivalence and can be thought of as an extension of regular equivalence to valued networks.

Then indirect approaches to blockmodeling are presented in Chapter 4. Several modifications are suggested for one of the indirect approaches, namely for REGE (White, 2005). These modifications are aimed to make this approach more similar (or compatible) with generalized blockmodeling and to make it more suitable for the type of networks where tie values are influenced by the ‘size’ of units (e.g., trade networks among counties).

Direct approach is introduced in Chapter 5. The most important direct approach (for the dissertation) is generalized blockmodeling, which is presented in Chapter 6. The notion of (generalized) blockmodeling type is introduced to distinguish among different approaches to generalized blockmodeling of binary and valued networks. The binary (generalized) blockmodeling (Doreian et al.: 2005), the type of criterion function that is used and its ideal blocks are presented in Chapter 6. Some ideas of Batagelj and Ferligoj (2000) for generalized blockmodeling of valued networks are also presented there. The topics covered in Chapter 6 are used as a foundation on which the approaches to generalized blockmodeling of valued networks are developed in Chapter 7.

The first of these approaches, the valued blockmodeling, is a straightforward generalization of the binary (generalized) blockmodeling as presented by Doreian et al. (2005). It could be argued that all features of binary blockmodeling can be generalized to valued blockmodeling\textsuperscript{123}. Binary blockmodeling is actually only a special case of valued network.

\textsuperscript{122} Where values on the ties of relations are measured at least on an interval scale.

\textsuperscript{123} Generalization to two-mode networks, other block types, etc.
blockmodeling. Therefore it is expected that for valued network where the measurement scale is at least interval, the valued blockmodeling would always be superior or at least equal to binary blockmodeling. This has proven to be true in examples in Chapter 9 and in most but not all simulations presented in Chapter 10.

We can say that binary blockmodeling treats ties either as relevant or as nonexistent (or irrelevant). Valued blockmodeling is similar, yet it assesses if a tie is closer to being relevant or closer to nonexistent. If this is all that is important and if we have an idea about the criteria for determining which ties are relevant, these two approaches are well suited for partitioning the network and determining types of blocks in the obtained blockmodel. They can however not distinguish between blocks that do not differ in the type of block (pattern of ties in a block), but only in the values of ties (the differences more subtle than the distinction between relevant and nonexistent ties) relevant for that block type. Consequently, such differences in tie values can also not be used to find partitions.

The second presented approach is the homogeneity approach. It addresses one of the main problems of valued and binary blockmodeling. When using valued blockmodeling, a threshold must be explicitly selected that tells us how strong a tie must be to be treated as relevant. Similar threshold must also be selected when using binary blockmodeling, although it is sometimes implicitly set to the minimum positive value (all ties that exist, meaning are higher as 0, are treated as relevant). However, the compatibility of some block types in homogeneity blockmodeling is questionable. In addition, the null block type is not really used in homogeneity blockmodeling (without some modifications), as it is seen as a special case of complete block type. Therefore, homogeneity blockmodeling is well suited for distinguishing empirical blocks based on tie values and finding partitions based on such differences, but it is less suited for distinguishing empirical block based on block types and finding partitions based on such distinctions. This is especially problematic, since it is also not well suited for distinguishing between null block type and other block types, the distinction that is of great importance in blockmodeling. The null block type is namely the only block type that does not induce a tie between equivalence classes (or clusters) in the reduced graph.

The third approach, implicit blockmodeling is developed on the ideas of Batagelj and Ferligoj (2000, 11-13) presented already at the end of Chapter 6. Implicit blockmodeling can distinguish empirical blocks both based on tie values and block types. However, due to the fact that it is heavily influenced by the block maximums, it often classifies blocks differently as it would be desired. As the partitions that it finds are heavily influenced by the classification of blocks, it often also leads to unsatisfactory partitions. This is especially problematic if block maximum normalization is also used. The partitions that it produces can be improved, but this improvement comes at the price of one of the main advantages of implicit blockmodeling, which is its ability to distinguish between null block type and other block types. Nevertheless, examples in Chapter 9 and simulations in Chapter 10 have shown that the implicit blockmodeling can produce good results. The approach is especially not suitable for use on networks where the distribution of the tie values has extreme values.
All suggested approaches are implemented in the blockmodeling package (Žiberna, 2007a) for statistical program and environment R (R Development Core Team 2006). These and some other approaches are then tested on four empirical and two constructed examples in Chapter 9. The emphasis is usually on the generalized blockmodeling approaches, especially those for blockmodeling of valued networks. The exceptions are two examples where different versions of the REGE algorithm are studied. Several properties or characteristics of the approaches or their implementations are learned or confirmed through these examples. The results that follow are mainly based on the empirical examples.

One of the problems that any researcher attempting to use blockmodeling approaches in real networks faces is which approach is the most suitable. Ideally, the researchers would select the approach whose definition of some equivalence or ideal blocks best matches the theoretical concept that they want to use to obtain partitions. However, this is rarely possible in a more exploratory analysis.

In general, generalized blockmodeling usually performs well, although binary blockmodeling sometimes provides less suitable partitions or larger number of partitions. One of the valued blockmodeling partitions is often selected as the most suitable one, but this might be partly due to the fact that several values of parameter $m$ are used and the best partitions are then selected. Currently, binary blockmodeling has a unique advantage in terms of implementation as it is the only approach implemented in a relatively fast program (in Pajek 1.11 (Batagelj and Mrvar, 2006)).

Homogeneity blockmodeling does not suffer these problems, but from the so-called ‘null block problem’. This brings two undesirable effects that are observed in the examples. The first is that homogeneity blockmodeling by itself cannot classify blocks as null as opposed to some other block type. The second effect is that homogeneity blockmodeling often produces less clear ‘null blocks’.

Implicit blockmodeling produces good partitions, if used without null blocks. If used in this way, ‘the null block problem’ applies also to implicit blockmodeling, although the effect that the null blocks are less clear is less serious. When used with null blocks, the partitions obtained are usually slightly worse and the blockmodels are inappropriate.

The indirect approaches perform relatively well in most examples, although usually not as well as the generalized blockmodeling approaches. In some cases, the indirect approaches are able to identify reasonable partitions at some, but not at all numbers of clusters in a partition. Due to the speed of their computation, their use is strongly supported at least as a preliminary step (to assess the appropriateness of the structural or regular partitions and of the number of clusters).

---

124 As mentioned above, homogeneity blockmodeling does not classify blocks as null. The classification is usually done on block averages or some similar statistics based on tie values in a block.
As mentioned, generalized blockmodeling usually performs better. However on networks where the tie values that are in some sense influenced by the ‘size’ of the units (e.g., trade networks among counties, carbon flow networks), a version of REGE, especially designed for such networks, provides good partitions. Generalized blockmodeling might provide reasonable partitions if applied to a suitably normalized network, but they are unable to provide such a partition when used simultaneously on row and column normalized matrices.

In two examples image graphs are constructed based on the partitions obtained. Several rules are used for obtaining these image graphs. The averaging rules based on sum of squares blockmodeling and valued blockmodeling probably produce the best image graphs. Those based on the sum of squares approach give more emphasis to tie values in the block (while taking into account the type of block), while those based on valued blockmodeling give more emphasis to the pattern of ties (while taking into account how far the tie values are from 0 and the from parameter $m$).

A relatively comprehensive simulations study where the proposed approaches are evaluated on generated valued regular networks is presented in Chapter 10. The simulations are done in two stages. These conclusions are mainly based on the results of Stage 2. The reason for that is that Stage 1 is primarily intended as a preliminary stage, as networks with only 11 units were analyzed in Stage 1, which is due to the characteristics of regular blocks not enough units to produce relevant results regarding regular equivalence.

The most surprising result is the relative effectiveness of the methods for structural equivalence on simulated networks based on $\max$-regular equivalence. The sum of squares blockmodeling according to structural equivalence (direct approach) performed especially well. They usually do not perform as well as methods for regular equivalence, but much better than expected.

The simulation results also show that the approaches developed in this dissertation, especially homogeneity and implicit blockmodeling, are better tools for identifying regular partitions in a valued network than the existing direct and indirect approaches. REGE performs well in all settings for which it was designed, that is in settings where the partition that is searched for is maximal regular, although the results are not as good if the partition that is searched for can be considered maximal regular only if the values are also taken into account. Implicit and especially homogeneity (with $\max$-regular blocks) blockmodeling are generally the best approaches.

Binary and valued blockmodeling perform relatively well in Stage 1, especially when the partition searched for is not maximal regular and no additional information in terms of tie values is provided. Unfortunately, they usually perform terribly in Stage 2. Especially in the case of binary blockmodeling, there are indications that the problem is in the search procedure as the correct partition usually has less inconsistency than the one found by a local search.

However, the poor performance only occurs when these two blockmodeling types are not used with pre-specified blockmodeling. When this additional information is used, they are usually the best or among the best approaches. The use of pre-specified blockmodeling also
considerably improves the results of implicit blockmodeling in settings where the partition that is searched for is not maximal regular.

When simulating networks based on the generalized equivalence, the generalized blockmodeling types show their potential, although some additional research is clearly needed at least for homogeneity blockmodeling.

In the final chapter, the proposed approaches are compared based on their theoretical characteristics and their performance in examples and simulation.

15.2 Slovene abstract (Slovenski povzetek)


Poleg posplošenega bločnega modeliranja omrežij z vrednostmi na povezavah v disertaciji obravnavam tudi ostale pristope za bločno modeliranje omrežij z vrednostmi na povezavah. Za enega od teh pristopov, natančneje za posredni pristop za bločno modeliranje v skladu z regularno enakovrednostjo, predlagam v disertaciji tudi nekaj sprememb. Osnovni cilj disertacije je ustvariti čim boljše pristope za bločno modeliranje omrežij z vrednostmi na povezavah glede na želene lastnosti iskanih rešitev.

15.2.1 Kratak uvod v posplošeno bločno modeliranje


Posplošeno bločno modeliranje uporablja neposreden pristop. V primerjavi z ostalimi neposrednimi pristopi je njegova glavna prednost prilagodljivost različnim tipom želenih rešitev in količini predhodnega znanja oziroma vedenja, ki ga želimo vključiti v proces bločnega modeliranja. Tako se lahko uporablja za iskanje razbitij v skladu z različnimi enakovrednostmi, med drugim tudi v skladu s posplošeno enakovrednostjo. Posplošena enakovrednost ni posebna vrsta enakovrednosti, ampak predstavlja bolj način za definiranje enakovrednosti ‘po meri’. Posplošeno enakovrednost definiramo z izborom dovoljenih vrst povezav med skupinami in znotraj njih ter po želji tudi z njihovim položajem v bločnem modelu. Posplošeno bločno modeliranje je bilo do sedaj razvito le za binarna in označena omrežja.

V tej disertaciji so razviti pristopi za posplošeno bločno modeliranje omrežij z vrednostmi na povezavah. Ti pristopi so še vedno del posplošenega bločnega modeliranja (Doreian in drugi, 2005) in uporabljajo tudi isto vrsto kriterijske funkcije. Najpomembnejša razlika med pristopi, ki so razviti v tej disertaciji in med posplošenim bločnim modeliranjem Doreiana in drugih (2005) je v tem, da je posplošeno bločno modeliranje Doreiana in drugih (2005) možno uporabiti le za analizo binarnih in označenih omrežjih, medtem ko so pristopi razviti v tej disertaciji posebej prilagojeni za analizo omrežij z vrednostmi na povezavah, kjer so te vrednosti merjene vsaj z intervalno mersko lestvico. V ta namen so predlagane vrste blokov, primerne za omrežja z vrednostmi na povezavah. Skupna lastnost vseh pristopov k posplošenem bločnem modeliranju pa je poleg uporabe iste vrste kriterijske funkcije tudi njihova sposobnost določiti lastnosti želenega razbitja z vrsto enakovrednosti ali s posplošeno enakovrednostjo. Kot je že bilo omenjeno, je posplošena enakovrednost definirana z izborom dovoljenih vrst povezav med in znotraj skupin ter po želji tudi z njihovim položajem v bločnem modelu.

15.2.2 Kratak pregled disertacije

Z namenom umestitve razvitih pristopov so v drugem in tretjem poglavju predstavljena omrežja, analiza omrežij, bločno modeliranje in enakovrednosti. Najpomembnejši doprinos v teh dveh poglavjih je razprava o enakovrednostih, primernih za omrežja z vrednostmi na povezavah. V tem sklopu je predlagana tudi nova vrsta enakovrednosti, $f$-regularna enakovrednost, ki predstavlja možno razširitev regularne enakovrednosti na omrežja z vrednostmi na povezavah.

V četrtem poglavju so obravnavani posredni pristopi k bločnem modeliranju. V tem poglavju je predstavljenih tudi več novih različic algoritma REGE (White, 2005), posrednega pristopa za bločno modeliranje v skladu z regularno enakovrednostjo. Nove različice tega algoritma so razvite z namenom narediti ta algoritem bolj ustrezno za uporabo skupaj s posplošenim bločnim modeliranjem ter z namenom prilagoditve tega algoritma za analizo omrežij, kjer so vrednosti na povezavah odvisne od ‘velikosti’ (ne njuno v dobesednem smislu) enot. Primer takih omrežij so na primer omrežja trgovanja med državami, kjer so vrednosti na povezavah merjene v denarni enoti.
Neposredni pristopi so predstavljeni v petem poglavju. Z vidika te disertacije najpomembnejši od teh, posplošeno bločno modeliranje, je predstavljen v šestem poglavju. Tu je vpeljan tudi pojem vrsta (posplošenega) bločnega modeliranja z namenom razlikovanja med različnimi pristopi k bločnem modeliranju binarnih omrežij in omrežij z vrednostmi na povezavah. Predstavljeni so binarno bločno modeliranje (Doreian in drugi: 2005), vrsta kriterijske funkcije in idealni bloki, ki jih uporablja binarno bločno modeliranje. Prav tako so predstavljeni tudi zametki posplošenega bločnega modeliranja za omrežja z vrednostmi na povezavah, kot sta jih razvila Batagelj in Ferligoj (2000). Teme, predstavljene v šestem poglavju, predstavljajo temelje, na katerih so razviti pristopi za bločno modeliranje omrežij z vrednostmi na povezavah v sedemem poglavju.

Kot prvi od teh pristopov je predstavljeno obteženo bločno modeliranje, ki je razmeroma enostavna razširitev binarnega (posplošenega) bločnega modeliranja (Doreian in drugi, 2005). Za ta pristop pokažem, da je mogoče nanj prenesti praktično vse lastnosti binarnega bločnega modeliranja. Binarno bločno modeliranje je pravzaprav poseben primer uteženega bločnega modeliranja. Na podlagi tega je mogoče pričakovati, da se bo na omrežjih, kjer so vrednosti na povezavah merjene vsaj na intervalni lestvici, obteženo bločno modeliranje vedno obneslo vsaj tako dobro kot binarno. To je potrjeno na primerih, predstavljenih v devetem poglavju, ne pa tudi v vseh simulacijah, prestavljenih v desetem poglavju.

Binarno bločno modeliranje obravnava (potencialne) povezave bodisi kot pomembne bodisi kot neobstoječe (oziroma nepomembne). Obteženo bločno modeliranje je na nek način podobno, vendar pa tudi upošteva, kako daleč je povezava od obeh skrajnih vrednost, torej kje na intervalu med neobstoječo (vrednost povezave 0) in pomembno (vrednost povezave enaka prametru $m$) je. Če je ta informacija zadostna za razlikovanje med bloki in če imamo podlago za odločitev, katere povezave so pomembne, sta ta dva pristopa, še posebej pa obteženo bločno modeliranje, zelo primerna za bločno modeliranje omrežij z vrednostmi na povezavah. Vendar pa nista sposobna razlikovati med bloki, ki se ne razlikujejo po vrsti bloka (vzorcu povezav v bloku), ampak po vrednosti povezav, kjer se razlike med vrednostmi povezav ne odražajo v razlikovanju med pomembnimi in neobstoječimi povezavami. Posledično takih razlik med vrednostmi na povezavah ne moremo uporabiti za iskanje primernih razbitij enot.

Drugi predstavljeni pristop oziroma vrsta bločnega modeliranja je homogeno bločno modeliranje. Ta pristop nima težav binarnega in uteženega bločnega modeliranja z razlikovanjem med bloki na podlagi vrednosti na povezavah. Prav tako pri uporabi tega pristopa ni potrebna opredelitev meje, ki nam pove, ali je povezava pomembna ali neobstoječa. Vendar pa je pri homogenem bločnem modeliranju vprašljiva kompatibilnost nekaterih vrst blokov. Poleg tega se znotraj homogenega bločnega modeliranja ne uporablja praznega bloka (vsaj v osnovni različici), saj je le-ta poseben primer praktično vseh ostalih vrst blokov, predvsem pa popolnega bloka. Posledično je homogeno bločno modeliranje

---

125 Prazen blok je blok, kjer ni povezav, popoln blok pa tisti, kjer so prisotne vse (možne) povezave.
zelo primerno za razlikovanje blokov na podlagi vrednosti na povezavah in iskanju razbitij enot na podlagi vrst blokov, torej na podlagi vzorca povezav v blokih. To predstavlja še posebno velik problem, saj homogeno bločno modeliranje ne more razlikovati niti med praznim blokom in drugimi vrstami blokov. To pa je najpomembnejša razlikovanje med vrstami blokov, saj le prazen block ne predstavlja povezava v bločnem modelu.

Tretji pristop, *implicitno* bločno modeliranje, je bil razvit na podlagi iztočnic Batagelj in Ferligoj (2000, 11-13), ki so bile predstavljene že na koncu šestega poglavja. Implicitno bločno modeliranje lahko teoretično razlikuje med bloki tako na podlagi vrst blokov kot tudi na podlagi vrednosti na povezavah. Vendar so bločni modeli, ki so rezultat tega pristopa, pod velikim vplivom največjih vrednost v blokih. Ti bločni modeli so ponavadi zelo drugačni, kot bi si želeli oziroma pričakovali glede na vrednosti povezav v blokih. Posledično tudi pogosto najde neprimerna razbitja enot v omrežju. Ta problem je še posebej velik, če je uporabljena normalizacija z največjo vrednostjo bloka. Ta razbitja je sicer mogoče izboljšati, a trenutno le za ceno ene izmed glavnih prednosti implicitnega bločnega modeliranja, sposobnosti, da razlikuje med praznim blokom in ostalimi vrstami blokov. Implicitno bločno modeliranje se je s takim popravkom dobro obneslo na primerih, predstavljenih v devetem poglavju, ter v simulacijah, prestavljenih v desetem poglavju. Vendar pa ta pristop ni primeren za analizo omrežij, kjer porazdelitev vrednosti na povezavah vsebuje izstopajoče vrednosti.

Vsi pristopi razviti v tej disertaciji so bili implementirani v paketu *blockmodeling* (Žiberna, 2007a), dodatku za statistični program in okolje R (R Development Core Team 2006). Ti in še nekateri drugi pristopi so preverjeni na štirih empiričnih in dveh umetnih primerih, ki so predstavljeni v devetem poglavju in simulacijah, prestavljenih v desetem poglavju. Vendar pa ta pristop ni primeren za analizo omrežij, kjer porazdelitev vrednosti na povezavah vsebuje izstopajoče vrednosti.

Z uporabo izbranih pristopov na prej omenjenih primerih je odkrito ali potrjeno kar nekaj značilnosti teh pristopov oziroma njihovih implementacij. V nadaljevanju predstavljeni rezultati temeljijo predvsem na štirih empiričnih primerih.

Eho prvih vprašanj, s katerim se srečamo, ko želimo z bločnim modeliranjem analizirati neko empirično omrežje, je, kateri pristop naj uporabimo. V idealnih okoliščinah bi izbrali pristop, katerega definicija enakovrednosti oziroma idealnih blokov najbolj ustreza teoretičnemu problemu, ki ga želimo rešiti. Žal pa pogosto problem ni dovolj natančno definiran, še posebej, če je analiza bolj raziskovalne (eksplorativne) narave. V tem primeru nam lahko pomagajo tudi spodaj navedene lastnosti, prednosti in slabosti pristopov.

V splošnem lahko trdimo, da so rezultati, pridobljeni s posplošenim bločnim modeliranjem, dobri. Razbitja enot pridobljena z binarnim bločnim modeliranjem sicer v nekaterih primerih niso najbolj primerna, ali pa je rezultat analize veliko število razbitij (tudi po več 100 razbitij, ki so optimalna glede na vrednost kriterijske funkcije). Kot najprimernejša rešitev je ponavadi izbrano razbitje, pridobljeno z uteženim bločnim modeliranjem, vendar pa je to lahko vsaj...
deloma tudi posledica dejstva, da je uporabljeno več različnih vrednosti za parameter $m$, nato pa izbrano najustreznejše razbijte. Vendar pa je potrebno omeniti, da ima trenutno binarno bločno modeliranje edinstveno prednost pred ostalimi pristopi posplošenega bločnega modeliranja. Je namreč edini pristop posplošenega bločnega modeliranja, ki je implementiran v Pajku (Batagelj in Mrvar, 2006), implementacija tu pa je mnogo (tudi 1000-krat) hitrejša od tiste v paketu blockmodeling.

Homogeno bločno modeliranje nima teh pomanjkljivosti. Njegova glavna pomanjkljivost je tako imenovani ‘problem praznega bloka’. Ta pomanjkljivost ima dve neželeni posledici, ki se kažeta tudi pri uporabi tega pristopa na primerih v devetem poglavju. Prva posledica je, da samo po sebi homogeno bločno modeliranje ne more ločiti med praznim blokom in ostalimi vrstami blokov. Druga je, da homogeno bločno modeliranje pogosto najde ‘prazne’ bloke, vendar pa vseeno vsebujejo več povezav kot prazni bloki, pridobljenimi z drugimi pristopi.

Implicitno bločno modeliranje ponavadi najde dobra razbitja enot, vendar pogosto le, če se uporablja brez praznih blokov. Če se uporablja na tak način, se ‘problem praznega bloka’ nanaša tudi na ta pristop, vendar pa imajo bloki, ki so podobni praznim, pri tem pristopu ponavadi vseeno manj močnejših povezav kot taki bloki, pridobljeni z homogenim bločnim modeliranjem. Kadar se implicitno bločno modeliranje uporablja z praznimi bloki, so pridobljena razbitja ponavadi slabša, bločni modeli pa neustrezni.

Rezultati posrednih pristopov so v večini primerov relativno dobri, vendar ponavadi ne tako dobi kot tisti, ki so bili pridobljeni s posplošenim bločnim modeliranjem. V nekaterih primerih so posredni pristopi sposobni identificirati ustrezen razbitja pri nekaterih, ne pa pri vseh izbranih številnih skupin. Vendar pa je zaradi hitrosti računanja njihova uporaba zelo priporočljiva, vsaj kot začetni korak v analizi (za oceno primernosti strukturne in regularne enakovrednosti in števil skupin).

Omenil sme že, da je posplošeno bločno modeliranje ponavadi dalo boljše rezultate kot posredni pristopi. Izjema so omrežja, kjer ‘velikost’ enot vpliva na vrednosti na povezavah (na primer pri omrežjih toka ogljika med živimi bitji). Tu različica algoritma REGE, ki je razvita posebej za taka omrežja, najde dobre rezultate. Posplošeno bločno modeliranje bi morda tudi lahko našlo primerna razbitja, če bi bilo uporabljeni na ustrezno normaliziranih omrežjih (kar pa ni bilo), pri sočasnem uporabi na vrsti in stolpčno normaliziranih matrikah pa takih razbitij ne najde.

V dveh primerih so izračunane tudi vrednosti na povezavah v bločnem modelu na podlagi večjega števila pravil. Pravila za računanje vrednosti teh povezav se nahajajo v podlagi homogenega bločnega modelirana na osnovi vseh kvadratov in uteženega bločnega modeliranja dajo najustreznejše vrednosti. Pravila na osnovi homogenega bločnega modelirana na osnovi vseh

---

126 Homogeno bločno modeliranje ne loči praznih blokov od preostalih. Tako tudi ne označi nobenega bloka kot ‘praznega’. S tem izrazom so tu označeni bloki, ki so podobni idealnim praznim blokom, oziroma bloki, ki imajo relativno malo (predvsem močnih) povezav.
kvadratov dajejo več poudarka vrednostmi na povezavah (ob upoštevanju vrste blokov), tista na podlagi uteženega bločnega modeliranja pa dajo več poudarka vzorcu povezav v blokih (ob upoštevanju, kako daleč so vrednosti povezav od 0 ali od parametra $m$).

V desetem poglavju je predstavljena razmeroma obsežna simulacijska študija, kjer je ovrednoteno delovanje predlaganih pristopov na uteženih, regularnih omrežjih. Simulacije so narejene v dveh fazah. V tem povzetku so predstavljeni predvsem rezultati druge faze. Prva faza je bila namenjena predvsem kot začetna faza, saj so omrežja z le 11 enotami premajhna za zanesljive rezultate na regularnih omrežjih, saj so majhni regularni bloki (kot posledica majhnih omrežij) razmeroma podobni polnim blokom.

Najbolj presenetljiv rezultat so razmeroma dobri rezultati metod na podlagi strukturne enakovrednosti na omrežjih, generiranih na podlagi maksimum-regularne enakovrednosti. Rezultati homogenega bločnega modeliranja na osnovi vsote kvadratov za strukturno enakovrednost (neposredni pristop) so še posebej dobri. Rezultati teh pristopov so sicer slabši kot tisti za regularno enakovrednost, vendar pa precej boljši, kot je bilo moč pričakovati.

Rezultati simulacij kažejo, da so pristopi, razviti v tej disertaciji, posebej homogeno in implicitno bločno modeliranje, boljši pristopi za iskanje regularnih razbitij v regularnih omrežjih z vrednostmi na povezavah kot obstoječi posredni in neposredni pristopi. Algoritmi REGE se dobro obnesejo v situacijah, za katere so razviti, to je v situacijah, kjer je iskano razbitje največje regularno razbitje. Vendar pa rezultati v celoti niso tako dobri, če je iskano razbitje največje regularno razbitje šele potem, ko upoštevamo tudi vrednosti na povezavah. V splošnem se v simulacijah najbolje obneseta implicitno in še posebej homogeno (z maksimum-regularnimi bloki) bločno modeliranje.

Binarno in obteženo bločno modeliranje se razmeroma dobro obneseta v prvi fazi, še posebej ko iskano razbitje ni največje regularno razbitje, če upoštevamo tudi vrednosti na povezavah. Žal pa v drugi fazi večinoma odpovesta. Še posebej rezultati za binarno bločno modeliranje kažejo, da je problem vsaj deloma v načinu iskanja ustrezne rešitve, saj ima pravilno razbitje (tisto, ki je uporabljeno pri generiranju omrežij) praviloma manjša odstopanja od idealne rešitve kot tisto razbitje, ki je najdeno z lokalno optimizacijo.

Vendar pa se binarno in obteženo bločno modeliranje slabo obneseta le, kadar se uporabita brez v naprej določenega bločnega modela. Kadar je bločni model v naprej določen, sta ta dva pristopa med najboljšimi pristopi. Uporaba v naprej določenega bločnega modela tudi znatno izboljša rezultate implicitnega bločnega modeliranja v situacijah, kjer iskano razbitje ni največje regularno razbitje.

Pri simulacijah na omrežjih, generiranih na podlagi posplošene enakovrednosti, posplošeno bločno modeliranje nakaže svoje potenciale, pa tudi potrebo po dodatnih raziskavah, še posebej v zvezi s homogenim bločnim modeliranjem.

V zadnjem poglavju so predlagani pristopi primerjeni in ocenjeni na podlagi njihovih teoretičnih lastnosti ter rezultatov na primerih in simulacijah.
16 POSPLOŠENO BLOČNO MODELIRANJE OMREŽIJ Z VREDNOSTMI NA POVEZAVAH (RAZŠIRJEN POVZETEK)


Poleg posplošenega bločnega modeliranja omrežij z vrednostmi na povezavah v disertaciji obravnavam tudi ostale pristope za bločno modeliranje omrežij z vrednostmi na povezavah. Za enega od teh pristopov, natančneje za posredni pristop za bločno modeliranje v skladu z regularno enakovrednostjo, predlagam v disertaciji tudi nekaj sprememb. Osnovni cilj disertacije je ustvariti čim boljše pristope za bločno modeliranje omrežij z vrednostmi na povezavah glede na želene lastnosti iskanih rešitev.

16.1 Omrežja

Omrežje je množica enot, na katerih je definirana vsaj ena relacija. Wasserman in Faust (1994: 3-4) trdita, da analiza omrežij temelji na predpostavki o pomembnosti povezav med enotami, ki medsebojno vplivajo druga na drugo.

Poznamo več vrst omrežij. Prva pomembna delitev je na egocentrična in popolna omrežja. Tehnike bločnega modeliranja, ki so predlagane v tej disertaciji, lahko uporabimo le na popolnih omrežjih, saj so le v teh omrežjih lahko definirani bloki. Blok je skupina vseh povezav med dvema skupinama akterjev (ali med akterji ene skupine) in je osnova sestavina bločnih modelov.

Omrežja delimo tudi glede na število različnih množic enot v omrežju. Enovrstna omrežja imajo le eno množico, dvovrstna pa dve. Obstajajo sicer tudi večvrstna omrežja, a tu je tudi povezava definirana med več kot dvema enotama. V disertaciji sem se ukvarjal le z enovrstnimi omrežji, vendar pa so metode razvite v disertaciji primerne tudi za dvovrstna omrežja.

Za namen te doktorske disertacije pa je še posebej pomembno, s kakšno mersko lestvico so merjene relacije ali relacija v omrežju. Posplošeno bločno modeliranje je bilo namreč do sedaj uporabljen metode omogočajo le analizo binarnih omrežij (kjer povezava ali obstaja ali pa ne) in označenih omrežij predznak (Doreian in drugi, 2005), v tem delu pa sem se ukvarjal predvsem z razširitvijo teh pristopov tudi na omrežja z (vsaj intervalnimi) vrednostmi na povezavah, se pravi na obtežena omrežja. Vendar pa sta tudi že pred tem Batagelj in Ferligoj (2000) predstavila ideje za posplošeno bločno modeliranje na obteženih omrežjih (merjenih z
intervalno mersko lestvico), ki pa še ni bila implementirana in preizkušena.

Z vidika pristopov k (še posebej posplošenem) bločnem modeliranju je pomembna tudi velikost omrežja. Večina pristopov, razvitih v tej disertaciji, je namreč primernih predvsem za majhna omrežja. To še posebej velja za pristope k posplošenemu bločnem modeliranju omrežij z vrednostmi na povezavah. Trenutna implementacija v paketu blockmodeling (Žiberna, 2007b) omogoča le analizo zelo majhnih omrežij (do 50 enot, do 20-30 v manj kot nekaj urah/dnevu), vendar pa bi tudi z boljšo implementacijo s temi pristopi analizirali le razmeroma majhna omrežja (nekaj 100 enot).

16.2 Uvod v bločno modeliranje

Bločno modeliranje pa je metoda za razvrščanje enot v omrežju v skupine ter ugotavljanje povezav med temi skupinami in znotraj njih. Bločno modeliranje razvršča enote v skupine na podlagi izbrane enakovrednosti in istočasno povezave med enotami v bloki, ki so definirani z skupinami enot, med katerimi so te povezave. Tudi če enakovrednost ni izbrana v naprej, ampak se postavlja druge kriterije kot osnovo za razvrščanje, dobljeno razbitje vedno določa enakovrednost.


V primeru obteženih omrežij se tej nedorečenosti dodajo še druge, ki so povezane z vrednostjo povezav. Postavi se vprašanje, katere povezave morajo imeti enake vrednosti, da se dve enoti lahko smatrata za enakovredni. Ker je razliko zelo težko razložiti z besedami, bom raje uporabil matematičen zapis. Če se osredotočimo na primer, kjer se potencialne povezave med parom enot obravnavajo ločeno, je prva, strožja definicija taka:
Naj bo \(\equiv\) relacija enakovrednosti na \(U\). Potem je \(\equiv\) is a *regularna enakovrednost* če in samo če za vsak par \(a, b \in U\), \(a \equiv b\) pomeni da:
1. za vsak \(r_{ai}\) obstaja tak \(r_{bj}\), da velja \(r_{bj} = r_{ai}\), kjer je \(i \equiv j\), \(i, j \in U\)
2. za vsak \(r_{ia}\) obstaja tak \(r_{ja}\) da velja \(r_{ja} = r_{ia}\), kjer je \(i \equiv j\), \(i, j \in U\).


Naj bo \(\equiv\) relacija enakovrednosti na \(U\). Potem je \(\equiv\) is a *regularna enakovrednost* če in samo če za vsak par \(a, b \in U\), \(a \equiv b\) pomeni da:
1. za vsak \(r_{ai}\) obstaja tak \(r_{bj}\), da velja \(r_{bj} \geq r_{ai}\), kjer je \(i \equiv j\), \(i, j \in U\)
2. za vsak \(r_{ia}\) obstaja tak \(r_{ja}\) da velja \(r_{ja} \geq r_{ia}\), kjer je \(i \equiv j\), \(i, j \in U\).

Razlika med obema definicijama je le v tem, da se pri drugi enačbi ('\(\equiv\)') iz prve definicije zamenjajo z znakom 'je večje ali enako' (\(\geq\)).

Različici teh dveh definicij, kjer pa se potencialni par povezav med parom enot obravnava skupaj, uporablja tudi REGGE (White, 1985a, 1985b) v svojih algoritmih za računanje podobnosti oz. različnosti v smislu regularne enakovrednosti.

Blok vsebuje vse potencialne povezave ali med enotami iz dveh skupin ali znotraj skupine. Razliko med obema definicijama je morda lažje ponazoriti z lastnostmi blokov, ki jih dovoljujeta. Prva definicija predvideva, da so v vsakem bloku v vseh vrsticah in v vseh stolpcih zastopane enake vrednosti, dovoljuje pa, da so njihove frekvence različne (a povsod ali enake ali večje kot 0). Druga, manj stroga definicija pa zahteva, da so v vseh vrsticah in v vseh stolpcih znotraj bloka enake le največje vrednosti. Zato jo lahko zapišemo tudi takole:

Naj bo \(\equiv\) relacija enakovrednosti na \(U\), ki ustreza razbitju \(C\). Potem je \(\equiv\) *regularna enakovrednost* če in samo če za vsak par \(a, b \in U\) in za vse \(X \in C\), \(a \equiv b\) pomeni da:
1. \(\max_{i \in X}(r_{ai}) = \max_{i \in X}(r_{bi})\)
2. \(\max_{i \in X}(r_{ia}) = \max_{i \in X}(r_{ib})\).

V disertaciji definiram tudi bolj splošno verzijo te enakovrednosti, \(f\)-regularno enakovrednost, ki jo definiram takole:

Naj bo \(\equiv\) relacija enakovrednosti na \(U\), ki ustreza razbitju \(C\). Potem je \(\equiv\) *\(f\)-regularna enakovrednost* (kjer \(f\) predstavlja izbrano funkcijo, na primer maksimum, povprečje, vsoto, itd.) če in samo če za vsak par \(a, b \in U\) in za vse \(X \in C\), \(a \equiv b\) pomeni da:
1. \(f_{i \in X}(r_{ai}) = f_{i \in X}(r_{bi})\)
2. \(f_{i \in X}(r_{ia}) = f_{i \in X}(r_{ib})\).

Za posplošeno bločno modeliranje je najpomembnejša posplošena enakovrednost. Posplošena enakovrednost ni posebna vrsta enakovrednosti, ampak predstavlja bolj način za definiranje enakovrednosti ‘po meri’. Posplošeno enakovrednost definiramo z izborom dovoljenih vrst povezav med skupinami in znotraj njih (vrst blokov) ter po želji tudi z njihovim položajem v bločnem modelu.

Želene lastnosti razbitja pa lahko definiramo tudi drugače kot z enakovrednostjo, vendar pa, kost smo že omenili, tudi če enakovrednost ni izbrana v naprej, dobljeno razbitje vedno določa enakovrednost. Tako so lahko želene lastnosti razbitja določene tudi preko kriterijske funkcije direktno (se pravi, brez da bi se poprej definiralo enakovrednost). To je še posebej pogosto pri posrednih pristopih.

16.3 Bločno modeliranje


16.3.1 Posredni pristopi


Kot smo že omenili, je pri posrednih pristopih osnova neka mera podobnosti ali različnosti, ki je kompatibilna z izbrano enakovrednostjo. Različnost d je kompatibilna z enakovrednostjo ≜, če in samo če za enoti a, b ∈ U, a ≡ b natanko takrat, ko je d(a, b) = 0 (Batagelj in drugi, 1992b: 72).

Vendar pa je moč ustrezeno mero podobnosti ali različnosti analitično izračunati (ang. closed form) le za strukturno enakovrednost. Batagelj in drugi (1992b: 70-73) so pokazali, da niso vse različnosti, ki so se pogosto uporabljale za iskanje strukturne enakovrednosti, kompatibilne z njo. Pokazali so, da so z njo kompatibilne le različnosti (ali podobnosti),
popravljene na tak način kot popravljena evklidska različnost (Burt in Minor, 1983 v Batagelj in drugi, 1992b:71).

Za regularno enakovrednost pa obstajajo algoritmi, ki v iteracijah izračunajo mere podobnosti in različnosti v smislu regularne enakovrednosti. Verziji algoritma REGE, ki ju je razvil White (1985a, 1985b) sta kompatibilni z dvema različicama definicije regularne enakovrednosti (ki se ločita na način obravnavanja vrednosti na povezavah, kot je omenjeno v prejšnjem podpoglavlju), ki obravnavata par potencialnih povezav med parom enot skupaj. Ker se v poslošenem bločnem modeliranju obravnavata ti dve povezavi ločeno, sem v disertaciji razvil verzije teh tega algoritma, kjer se ti dve povezavi obravnavata ločeno (neodvisno).

Nordlund (2007, 61) je predstavil postopek za identifikacijo regularni blokov v omrežjih, kjer so vrednosti na povezavah odvisne od ‘velikosti’ (ne njuno v dobesednem smislu) enot. Primer takih omrežij so na primer omrežja trgovanja med državami, kjer so vrednosti na povezavah merjene v denarni enoti. Na podlagi teh idej sem razvil verzije algoritma REGE, ki je primeren za iskanje enakovrednosti na podlagi vzorca povezav v teh omrežjih.

Brandes in Lerner (2004) sta predlagalo novo skupino mer podobnosti za enote v neusmerjenih omrežjih, to je strukturno podobnost. Strukturne podobnosti merijo podobnost med enotami v omrežjih na podlagi strukturnih lastnosti. Te lastnosti so lahko centralnost, položaj na osi center-periferija, povezane skupine, nasprotuječe si skupine, itd. Kasneje sta (Brandes in Lerner, 2005) to mero razširila tudi na usmerjena omrežja. Strukturne podobnosti predstavljajo zanimiv razvoj na področju mer podobnosti med enotami omrežja, vendar pa je potrebno bolje raziskati pomen teh mer (kaj točno merijo), še posebej v usmerjenih omrežjih.

16.3.2 Neposredni pristopi


Poleg teh pristopov pa poznamo tudi stohastično bločno modeliranje (Snijders in Nowicki, 1997; Nowicki in Snijders, 2001). Stohastično bločno modeliranje tudi omogoča analiziranje omrežj z vrednostmi na povezavah, kjer pa so te vrednosti merjene le z nominalno lestvico. Bistvo tega pristopa je v predpostavki, da je verjetnost za povezavo pri binarnih omrežjih oziroma porazdelitvena funkcija vrednosti na povezavi (kjer vrednost 0 pomeni, da ni
povezave) pri omrežjih z vrednostmi na povezavah odvisna le od tega, katerima skupinama pripadata začetna in končna enota povezave.

16.4 Posplošeno bločno modeliranje

Doreian in drugi (2005) ločijo dve vrsti pristopov k bločnemu modeliranju. Prvo vrsto pristopov imenujejo klasični pristopi k bločnemu modeliranju, drugo pa posplošeno bločno modeliranje. Klasičnim pristopom k bločnemu modeliranju pripisujejo tri lastnosti:
1. uporabljen je posreden pristop,
2. uporabljena je strukturna ali regularna enakovrednost in
3. malo ali celo nič pozornosti je namenjeno merjenju, kako dobro dobljeni model ustreza omrežju.

Ti pristopi, pa tudi nekateri drugi so bili že opisani v prejšnjem podpoglavju.

Posplošeno bločno modeliranje definirajo ravno nasprotno:
1. uporabljen je neposreden pristop,
2. namesto z enakovrednostjo pa želene lastnosti določijo s pomočjo dovoljenih vrst blokov (in včasih tudi njihovih pozicij v bločnem modelu), kar omogoča veliko več raznovrstnosti in natančnosti, in
3. mera ustreznosti dobljenega bločnega modela glede na omrežje je vdelana v sam proces iskanja rešitve.


Njihov pristop (posplošeno bločno modeliranje) je implementiran tudi v programu Pajek (Batagelj in Mrvar, 2004), še z nekaj dodatnimi možnostmi (na primer tudi za dvovrstna omrežja) pa v programu Model 2 (Batagelj, 1996), ki je del paketa STRAN (Batagelj, 1991). Kot je že bilo omenjeno, posplošeno bločno modeliranje uporablja neposreden pristop. V primerjavi z ostalimi neposrednimi pristopi je njegova glavna prednost prilagodljivost različnim tipom želenih rešitev in količini predhodnega znanja oziroma vedenja, ki ga želimo vključiti v proces bločnega modeliranja. Tako se lahko uporablja za iskanje razbitij v skladu z različnimi enakovrednostmi, med drugim tudi v skladu s posplošeno enakovrednostjo.

127 Posreden pristop pomeni, da pristop ne išče razbitja akterjev v omrežju neposredno, ampak najprej izračuna matriko različnosti ali podobnosti med akterji, nato pa se na tej matriki uporabi ena od metod razvrščanja v skupine (Doreian in drugi, 2005).

128 Kljub temu da avtorji izrecno omenjajo le ti dve enakovrednosti, menim, da bi temu pogoju zadostila tudi uporaba katere koli enakovrednosti, na primer avtomorfne enakovrednosti (več o njej na primer v Everett in Borgatti, 2002)
Posplošena enakovrednost ni posebna vrsta enakovrednosti, ampak predstavlja bolj način za definiranje enakovrednosti ‘po meri’. Posplošeno enakovrednost definiramo z izborom dovoljenih vrst povezav med skupinami in znotraj njih ter po želji tudi z njihovim položajem v bločnem modelu. Posplošeno bločno modeliranje je bilo do sedaj razvito le za binarna in označena omrežja.

V tej disertaciji so razviti pristopi za posplošeno bločno modeliranje omrežij z vrednostmi na povezavah. Ti pristopi so še vedno del posplošenega bločnega modeliranja (Doreian in drugi, 2005) in uporabljajo tudi isto vrsto kriterijske funkcije. Najpomembnejša razlika med pristopi, ki so razviti v tej disertaciji in med posplošenim bločnim modeliranjem Doreiana in drugih (2005) je v tem, da je posplošeno bločno modeliranje Doreiana in drugih (2005) možno uporabiti le za analizo binarnih in označenih omrežjih, medtem ko so pristopi razviti v tej disertaciji posebej prilagojeni za analizo omrežij z vrednostmi na povezavah, kjer so te vrednosti merjene vsaj z intervalno mersko lestvico. V ta namen so predlagane vrste blokov, primerne za omrežja z vrednostmi na povezavah. Skupna lastnost vseh pristopov k posplošenem bločnem modelirjanju pa je poleg uporabe iste vrste kriterijske funkcije tudi njihova sposobnost določiti lastnosti želenega razbitja z vrsto enakovrednosti ali s posplošeno enakovrednostjo.

V disertaciji sem vpeljal tudi pojem vrsta (posplošenega) bločnega modeliranja z namenom razlikovanja med različnimi pristopi k bločnem modelirjanu binarnih omrežij in omrežij z vrednostmi na povezavah. Ti pristopi so binarno, obteženo, homogeno in implicitno bločno modeliranje in so opisani v nadaljevanju.

16.4.1 Kriterijska funkcija

Vsi ti pristopi pa imajo veliko skupnega, predvsem pa uporabljajo enako vrsto kriterijske funkcije in omogočajo uporabo posplošene enakovrednosti. Kriterijska funkcija, ki se uporablja v posplošenem bločnem modelirjanju, je definirana kot seštevek odstopanj oziroma odmikov empiričnega omrežja glede na izbrano razbitje. Idealna omrežja so definirana z dovoljenimi tipi blokov in izbranim razbitjem. Le ti so lahko definirani za vsak položaj v bločnem modelu posebej ali za vse skupaj. Dejansko ta odstopanja računamo za vsak empirični blok posebej tako, da izračunamo odstopanje tega empiričnega bloka od idealnih blokov za tipov blokov, dovoljenih za ta empirični blok (glede na njegov položaj v bločnem modelu). Najmanjše izračunano odstopanje predstavlja odstopanje tega empiričnega bloka od idealnega. Nato te vrednosti seštejemo za vse empirične bloke in dobimo skupno odstopanje omrežja. Različne vrste posplošenega bločnega modeliranja se razlikujejo po definiciji idealnih tipov blokov in po načinu računanja odstopanja empiričnih blokov od idealnih.

Za bolj formalen zapis moramo najprej vpeljati nekaj ozna:

- \( N = (U, R) \) predstavlja omrežje, kjer je \( U \) množica vseh enot \( U = (u_1, u_2, \ldots, u_n) \) in \( R \) relacija med temi enotami \( R \subseteq U \times U \)
V posplošnem bločnem modeliranju relacijo $R$ ponavadi predstavimo z matriko $R$, kjer $r_{ij}$ presečila vrednost povezave od enote $i$ do enote $j$, če ta obstaja, in ima vrednost 0, če ne obstaja.

$C = \{C_1, C_2, \ldots, C_n\}$ razbitje množice $U$. $\Phi$ je množica vseh možnih razbitij. Razbije $C$ razvrstana tudi relacijo $R$ v empirične bloke $R(C_i, C_j) = R \cap C_i \times C_j$. Vsak tak blok sestavljajo enote iz skupin $C_i$ in $C_j$ ter vse povezave iz skupine $C_i$ do skupine $C_j$. Če je $i = j$, potem blok $R(C_i, C_j)$ imenujemo diagonalni blok.

Naj $T(C_i, C_j)$ označuje množico vseh idealnih blokov (za vse dovoljene tipe blokov), ki ustrezajo bloku $R(C_i, C_j)$.

$w(T)$ je utež bloka tipa $T$. Ponavadi so uteži vseh blokov enake 1, vendar pa imajo lahko različni tipi blokov tudi različne uteži.

Odstopanje od bloka od nekega tipa $\delta(R(C_i, C_j), T)$ meri odstopanje empiričnega bloka $R(C_i, C_j)$ od idealnega bloka $T \in T(C_i, C_j)$.

Odstopanje bloka $p(C_i, C_j)$ pa nato izračunamo kot $p(C_i, C_j) = \min_{T \in T(C_i, C_j)} \left( w(T) \delta(R(C_i, C_j), T) \right)$.

Skupno odstopanje omrežja $P(C)$ od idealnega omrežja glede na razbitje $C$ lahko izrazimo kot vsoto odstopanj vseh blokov: $P(C) = \sum_{C_i, C_j \in C} p(C_i, C_j)$.

Kot je že bilo omenjeno, idealne bloke $T$ in način računanja odstopanj od njih $\delta(R(C_i, C_j), T)$ določa vrsta posplošnega bločnega modeliranja.

16.4.2 Binarno bločno modeliranje

Binarno bločno modeliranje je posplošeno bločno modeliranje za binarna omrežja. To pomeni, da če želimo z njim analizirati obtežena omrežja, moramo iz njih najprej ustvariti binarna. To pomeni, da moramo vrednosti na povezavah rekodirati bodisi v 0, bodisi v 1.

Kot smo omenili, vse vrste bločnega modeliranja uporabljajo tip kriterijske funkcije, predstavljen v prejšnji točki. Tisto, kar loči vrste bločnega modeliranja med seboj so definicije idealnih blokov in način računanja odstopanj od njih. Opisi oziroma definicije večine idealnih blokov za vse vrste bločnega modeliranja so prikazani v Tabeli 14.1. Definicije za binarno bločno modeliranje so prikazane v drugem stolpcu. Tudi tu lahko vidimo, da so bloki definirani za vrednosti 0 in 1. Za uporabo tega bločnega modeliranja pa poleg teh definitij idealnih blokov rabimo tudi način računanja odstopanj od njih. V disertaciji predstavljamo formule za izračun teh odstopanj za vse tipe blokov, tu pa opisujem le način računanj teh odstopanj. V osnovi se odstopanja računajo kot število vrednosti, ki bi jih bilo potrebno spremeniti. Da pa so bloki med seboj bolj enakovredni, pri tistih z manj strogimi zahtevami, se pravi kjer zahtevamo, da nekaj velja samo za eno vrstico ali stolpec ali samo za en element v vrstici ali stolpcu, ta odstopanja pomnožimo za številom vrstic, stolpcev ali elementov v
njih (odvisno od pogoja). Pri večini blokov (izjema je le gostotni in bloki na diagonali) je največja možna vrednost tega odstopanja enaka številu celic v bloku.

16.4.3 Obteženo bločno modeliranje

Kot prvi od novih pristopov je predstavljeno obteženo bločno modeliranje, ki je razmeroma enostavna razširitev binarnega bločnega modeliranja (Doreian in drugi, 2005). Za ta pristop pokažem, da je mogoče nanj prenesti praktično vse lastnosti binarnega bločnega modeliranja. Binarno bločno modeliranje je pravzaprav poseben primer uteženega bločnega modeliranja. Na podlagi tega je mogoče pričakovati, da se bo na omrežjih, kjer so vrednosti na povezavah merjene vsaj na intervalni lestvici, obteženo bločno modeliranje vedno obneslo vsaj tako dobro kot binarno. To je potrjeno na primerih, predstavljenih v devetem poglavju, ne pa tudi v vseh simulacijah, prestavljenih v desetem poglavju.

Kot smo že omenili, binarno bločno modeliranje obravnava (potencialne) povezave bodisi kot pomembne bodisi kot neobstoječe (ozoroma nepomembne). Obteženo bločno modeliranje je na nek način podobno, vendar pa tudi upošteva, kako daleč je povezava od obeh skrajnih vrednosti, torej kje na intervalu med neobstojoče (vrednost povezave 0) in pomembno (vrednost povezave enaka prametru $m$) je. Med tem ko smo to, katere povezava je pomembna določali z mejo, ki je določala, katere vrednosti se rekodirajo v 1 (in katere v 0), to določamo z parametrom $m$. V obteženem bločnem modeliranju je povezava pomembna, če imam vrednost vsaj $m$, je nepomembna, če ima vrednost 0, če je pa nekje vmes, pa upoštevamo, kako blizu je obema ekstremoma.

Če je taka informacija zadostna za razlikovanje med bloki in če imamo podlago za odločitev, katere povezave so pomembne, sta ta dva pristopa, še posebej pa obteženo bločno modeliranje, zelo primerna za bločno modeliranje omrežij z vrednostmi na povezavah. Vendar pa obe vrsti bločnega modeliranja nista sposobni razlikovati med bloki, ki se ne razlikujejo po vrsti bloka (vzorcu povezav v bloku), ampak po vrednosti povezav, kjer so razlike med vrednostmi povezav ne odražajo v razlikovanju med pomembnimi in neobstoječimi povezavami. Posledično takih razlik med vrednostmi na povezavah ne moremo uporabiti za iskanje primernih razbitij enot.

Tako kot za binarno so tudi za obteženo bločno modeliranje definicije tipov blokov predstavljene v Tabeli 14.1 in sicer v tretjem stolpcu. Kot smo omenili že zgoraj, so tu definicije zelo podobne tistim za binarno bločno modeliranje, le da so pogoji, da mora nekaj biti 1 zamenjani s pogojem, da mora biti nekaj vsaj $m$. Pri binarnem bločnem modeliranju smo odstopanje empiričnih od idealnih blokov računali kot število celic, ki jim moramo spremeniti vrednost, da so izpolnjeni pogoji idealnih blokov. Pri obteženem bločnem modeliranju je postopek podoben, le da ne štejemo le, koliko celic moramo spremeniti, da zadostimo pogoje, ampak tudi za koliko jih moramo spremeniti. Enako kot pri binarnem bločnem modeliranju pa tudi tu pri manj strožjih zahtevah, torej pri takih, kjer zahtevamo, da nekaj velja samo za eno vrstico, stolpec ali element v vrstici ali stolpcu, ta odstopanja pomnožimo za številom vrstic, stolpcev ali elementov v njih (odvisno od pogoja).
Tabela 16.1: Definicije idealnih blokov za vse vrste bločnega modeliranja

<table>
<thead>
<tr>
<th>Idealni bloki in ‘oznaka’</th>
<th>binarno bločno modeliranje(^{129})</th>
<th>obteženo bločno modeliranje</th>
<th>homogeno in implicitno bločno modeliranje</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prazen ‘null’</td>
<td>vse celice 0 *</td>
<td>vse celice 0 *</td>
<td>vse celice 0 *</td>
</tr>
<tr>
<td>Polni ‘com’</td>
<td>vse celice 1 *</td>
<td>vse celice vsaj m *</td>
<td>vse enako **</td>
</tr>
<tr>
<td>Vrstično-dominantni ‘rdo’</td>
<td>obstaja vrstica samih 1 *</td>
<td>obstaja vrstica, kjer so vse celice vsaj m **</td>
<td>obstaja vrstica, kjer so vse celice enake **</td>
</tr>
<tr>
<td>Stolpčno-dominantni ‘cdo’</td>
<td>obstaja stolpec samih 1 *</td>
<td>obstaja stolpec, kjer so vse celice vsaj m **</td>
<td>obstaja stolpec, kjer so vse celice enake **</td>
</tr>
<tr>
<td>Vrstično-(f-)regularni ‘rre’</td>
<td>obstaja vsaj ena 1 v vsaki vrstici</td>
<td>vrednost funkcije ( f ) na vsaki vrstici je vsaj m</td>
<td>maksimum bloka se natanko enkrat pojavi v vsaki vrstici, vse ostale vrednosti so 0</td>
</tr>
<tr>
<td>Stolpčno-(f-)regularni ‘cre’</td>
<td>obstaja vsaj ena 1 v vsakem stolpcu</td>
<td>vrednost funkcije ( f ) na vsakem stolpcu je vsaj m</td>
<td>maksimum bloka se natanko enkrat pojavi v vsakem stolpcu, vse ostale vrednosti so 0</td>
</tr>
<tr>
<td>(f-) regularni ‘reg’</td>
<td>obstaja vsaj ena 1 v vsaki vrstici in vsakem stolpcu</td>
<td>vrednost funkcije ( f ) na vsaki vrstici in vsakem stolpcu je vsaj m</td>
<td>vrednost funkcije ( f^{*} ) je po na vseh vrsticah enaka *</td>
</tr>
<tr>
<td>Vrstično – funkcijski ‘rfn’</td>
<td>obstaja natanko ena 1 v vsaki vrstici</td>
<td>vsaki vrstici obstaja natanko ena 1 neničelna celica z vrednostjo vsaj m</td>
<td>vrednost funkcije ( f^{*} ) je po na vseh stolpcih enaka *</td>
</tr>
<tr>
<td>Stolpčno – funkcijski ‘cfn’</td>
<td>obstaja natanko ena 1 v vsakem stolpcu</td>
<td>vsakem stolpcu obstaja natanko ena 1 neničelna celica z vrednostjo vsaj m</td>
<td>vrednost funkcije ( f^{*} ) je po na vseh vrsticah in vseh stolpcih enaka *</td>
</tr>
</tbody>
</table>

Legenda:

* - na diagnali (v diagonalnih blokih) so dovoljene izjeme – kakšne točno so lahko izjeme je odvisno od vrste bločnega modela
* - možen je dodatni pogoj, da vrednosti, ki so enake, ne smejo biti enake 0 – ta pogoj se v originalni različici pojavlja (uporablja) pri implicitnem, ne pa tudi pri homogenem bločnem modeliranju
* - pri implicitnem bločnem modeliranju je funkcija \( f \) vedno maksimum
* - pri homogenem bločnem modeliranju obstaja več verzij definicij vrstično- in stolpčno-dominantnih blokov – predstavljena je le ena izmed njih

\(^{129}\) po Doreian in drugi (2005: 223)

V splošnem največje odstopanje od pogoja, da mora biti neka vrednost enaka 0, ni določeno. To tudi pomeni, da največje odstopanje od idealnih blokov, ki vsebujejo tak pogoj, ni omejeno. Ta problem lahko rešimo tako, da pred analizo knjimo omrežje pri vrednosti parametra $m$. To pomeni, da vse vrednosti, večje kot $m$, rekodiramo (spremenimo) v vrednost parametra $m$.

16.4.4 Homogeno bločno modeliranje

Drug pristop oziroma vrsta bločnega modeliranja je homogeno bločno modeliranje. Ta pristop nima težav binarnega in uteženega bločnega modeliranja z razlikovanjem med bloki na podlagi vrednosti na povezavah. Prav tako pri uporabi tega pristopa ni potrebna opredelitev meje, ki nam pove, ali je povezava pomembna ali neobstoječa. Vendar pa je pri homogenem bločnem modeliranju vprašljiva kompatibilnost nekaterih vrst blokov. Poleg tega se znotraj homogenega bločnega modeliranja ne uporablja praznega bloka (vsaj v osnovni različici), saj je le-ta poseben primer praktično vseh ostalih vrst blokov, predvsem pa popolnega bloka. Posledično je homogeno bločno modeliranje zelo primerno za razlikovanje blokov na podlagi vrednosti na povezavah in iskanju razbitih enot na podlagi razlikovanja, manj pa je primerno za razlikovanje blokov in iskanje razbitih enot na podlagi vrst blokov, torej na podlagi vzorca povezav v blokih. To predstavlja še posebno velik problem, saj homogeno bločno modeliranje ne more razlikovati niti med praznim blokom in drugimi vrstami blokov. To pa je najpomembnejše razlikovanje med vrstami blokov, saj le prazen blok ne predstavlja povezave v bločnem modelu.

Idealni bloki so tudi za homogeno bločno modeliranje opisani v Tabeli 14.1. Za vrstično- in stolpčno-dominantne bloke je predstavljena le ena izmer treh verzij, predstavljene v disertaciji. Drugi dve se razlikujeta po tem, da določata, da je dominanta vrstica ali stolpec tista, ki je bodisi tista z največjo srednjo vrednostjo (prva alternativa), bodisi tista, ki ima srednjo vrednost najbolj različno od ostalih ( druga alternativa). Kot vidimo, so vsi bloki definirani tako, da morajo biti določene vrednosti enake, nekje pa je tudi določeno, da morajo biti enake 0. Odstopanja od te enakosti predstavljajo odstopanja empiričnih blokov od idealnih, merimo pa jih z neko mero variabilnosti. V disertaciji predlagam dve merni variabilnosti, ki posledično vodita do dveh podvrst homogenega bločnega modeliranja. Prva je vsota kvadratnih odklonov, druga pa vsota absolutnih odklonov. Če je določeno, katerim vrednostim morajo biti vrednosti enake (recimo 0 pri praznih blokih ali kaki drugi vrednosti pri uporabi v naprej določenega bločnega modela), se odkloni računajo od te vrednoti, drugače pa od take, da bodo odkloni najmanjši. To pomeni o aritmetične sredine pri vsoti kvadriranih odklonov in od mediane pri vsoti absolutnih odklonov. Podobno kot pri prejšnjih
dveh vrstah bločnega modeliranja odklone pri manj strogih pogojih pomnožimo z ustreznim številom (številom vrstic, stolpcev ali elementov v stolpcih ali vrsticah, odvisno od pogodja).


Drugi problem, ki se pojavlja pri homogene bločnem modeliranju, je problem praznega bloka. Ta problem je posledica tega, da je prazen blok pri homogenem bločnem modeliranju primer praktično vseh ostalih vrst blokov, predvsem pa polnega bloka. Zato tudi ni sposobno

---

130 V Tabeli 14.1 je označeno, da je definicije blokov za homogeno bločno modeliranje mogoče popraviti tako, da se pri vseh razen pri praznem bloku izrecno zahteva, da vrednosti, ki so enake, niso enake 0. V tem primeru
ločiti tega bloka od ostalih. Ponavadi se pri homogenem bločnem modeliranju naknadno izračuna neko statistiko na blokih, s pomočjo katere se določi, kateri bloki se klasificirajo kot prazne. Vendar pa so tako klasificirali bloki pogosto manj čisti, se pravi imajo več in močnejše povezave, kot podobni bloki najdeni z drugimi vrstami bločnega modeliranja.

Odstopanja so sicer definirana tudi za prazen blok, vendar pa je posebej, kadar uporabljamo vsoto kvadratov kot mero variabilnosti, ta blok lahko izbran le, kadar je blok idealen prazen blok. Prazne bloke torej lahko uporabimo v naprej določenem bločnem modelu, vendar pa so v slabšem položaju.

V disertaciji predlagam tudi možne rešitve za ta problem praznega bloka, ki pa še niso bile preizkušene in implementirane. Ena od možnih rešitev, ki je predvsem primerna za bločno modeliranje z v naprej določenim bločnim modelom, je da tudi za druge tipe blokov določimo, od katere vrednosti naj se računanja odkloni. Vendar pa je to v večini primerov preveč restrikтивno. Druga možnost je, da se za preostale tipe blokov (se pravi raznoprazenega), določi minimalno vrednost, od katere se lahko računajo odkloni. Če je vrednost, ki bi dala najmanjše odklone večja od te vrednosti, se pri računanju odklonov uporabi ta idealna vrednost, če pa je manjša (ali enaka) kot minimalna vrednost, pa se računajo odkloni od minimalne vrednosti. Še ena možna rešitev je tudi uporaba homogenega bločnega modeliranja skupaj se še kako drugo vrsto bločnega modeliranja, ki pa tudi ni brez težav.

Ima pa to, da je prazen blok poseben primer vseh ostalih blokov tudi eno pozitivno posledico. Namreč, v primeru, da ne bi bil, bi bilo tudi pri uporabi f-regularnih, vrstično-f-regularnih in stolpčeno-f-regularnih blokov zraven uporabiti tudi prazen blok, kar bi posledično lahko povzročilo problem z kompatibilnostjo blokov. Ker pa je prazen blok poseben primer vseh ostalih, torej tudi teh, to ni potrebno.

16.4.5 Implicitno bločno modeliranje

Tretji razvit pristop, *implicitno* bločno modeliranje, je bil razvit na podlagi iztočnic Batagelja in Ferligoj (2000, 11-13). Implicitno bločno modeliranje lahko teoretično razlikuje med bloki tako na podlagi vrst blokov kot tudi na podlagi vrednosti na povezavah. Vendar so bločni modeli, ki so rezultat tega pristopa, pod velikim vplivom največjih vrednosti v blokih. Ti bločni modeli so ponavadi zelo drugačni, kot bi si želeli oziroma pričakovali glede na vrednosti povezav v blokih. Posledično tudi pogosto najde neprimerna razbitja entot v omrežju. Ta problem je še posebej velik, če je uporabljena normalizacija z največjo vrednostjo bloka. Ta razbitja je sicer mogoče izboljšati, a trenutno le za ceno ene izmed glavnih prednosti implicitnega bločnega modeliranja, sposobnosti, da razlikuje med praznim blokom in ostalimi vrstami blokov. Implicitno bločno modeliranje se je s takim popravkom dobro obneslo na primerih, predstavljenih v devetem poglavju, ter v simulacijah, prestavljencih

__________

prazen blok ni posebne primer ostalih.
v desetem poglavju. Vendar pa ta pristop ni primeren za analizo omrežij, kjer porazdelitev vrednosti na povezavah vsebuje izstopajoče vrednosti.

Idealni bloki so, razen manjših podrobnosti (glej * v tabeli), enaki kot tisti za homogeno bločno modeliranje in so zato predstavljeni skupaj z njimi v četrtem stolpcu v Tabeli 14.1. Vendar pa se bistveno razlikuje način računanja odstopanja empiričnih blokov od idealnih. Način odstopanja je zelo podoben tistem, uporabljenemu pri obteženem bločnem modeliranju, le da vrednost parametrov m nadomestimo z največjo vrednostjo bodisi bloka (običajno), bodisi le dominante vrstice ali stolpca (pri dominantnih blokah). Kot smo omenili, je to zelo podobno situaciji pri obteženem bločnem modeliranju. Pri obteženem bločnem modeliranju je odstopanje seštevek vrednosti, za kolikor moramo spremeniti vrednosti v celicah, da zadostimo pogojem, torej da je vrednost vsaj $m$ ali da je vrednost enaka 0. Pri manj strožjih zahtevah (blokah), torej pri takih, kjer zahtevamo, da nekaj velja samo za eno vrstico, stolpec ali element v vrstici ali stolpku, ta odstopanja pomnožimo za številom vrstic, stolpcev ali elementov v njih (odvisno od pogoja). Pri implicitnem bločnem modeliranju računamo odstopanja na enak način, le da pogoj, da mora biti vrednost enaka ali večja kot $m$, zamenjamo s pogojem, da mora biti vrednost enaka (večja ne more biti) bodisi največja vrednosti v bloku (pri večini blokov) bodisi največji vrednosti v vrstici ali stolpcu (pri dominantnih blokah).

Batagelj in Ferligoj (2000: 13) poleg tega predlagata tudi normalizacijo tako izračunanih odstopanj z uporabljeno največjo vrednostjo (maksimum normalizacija) in z velikostjo bloka. Vendar pa na podlagi ugotovitev v svoji disertaciji ti dve normalizaciji, še posebej pa maksimum normalizacijo odsvetujem. Ta namreče poslabša posledice problema, ki je opisan v nadaljevanju.

Pri implicitnem bločnem modeliranju namreč tip bloka določamo relativno glede na največjo vrednost v bloku (oz. pri dominantnem bloku glede na največjo vrednost v dominantni vrstici ali stolpcu), od teh vrednosti pa običajno (razen seveda pri praznem bloku) računamo odstopanja. To tudi pomeni, da v primeru, da imamo samo eno ekstremno vrednost v bloku, ta blok zelo močno odstopa od na primer polnega in regularnega bloka. Primer lahko vidimo na Sliki 14.1. Blok $R(1, 2)$ (levo spodaj) ima same enice in eno trojko. Če dovolimo le polne in prazne bloke, je ta blok označen (glede na implicitno bločno modeliranje) kot prazen blok, saj tista trojka povzroči, da se vsa odstopanja od idealnega bloka računajo od trojke. Med tem pa je blok $R(2, 1)$ (desno zgornj) označen kot poln blok, ker ima vse vrednosti enake in različne od 0 in je tako idealen polen blok. Torej je blok $R(1, 2)$ označen kot prazne, medtem ko je blok $R(2, 1)$, ki ima vse vrednosti enake ali manjše kot $R(1, 2)$ označen kot poln blok. To običajno ni zaželeno. Problem se ne pojavlja le pri primerjavi praznega in polnega bloka, ampak tudi pri ostalih blokih. Posledično daje implicitno bločno modeliranje pogosto tudi neprimerna razbijta, povsem neprimerni pa so običajno bločni modeli. Kot smo že omenili, je ta problem še večji, če se uporabi normalizacija z največjo vrednostjo. Če se ta normalizacija ne uporablja in pri poljubnem bloku večamo njegovu največjo vrednost, se odstopanja tega bloka do vseh idealnih blokov (izjema sta lahko dominantna bloka, ki se osredotočata samo na eno vrstico oz. stolpec) večajo, vendar najpočasneje pri praznem bloku. Če pa isto
počnemo ob uporabi maksimum normalizacije, se pri takem početju odstopanje tega empiričnega bloka (če ima poleg največje vrednosti še kako neničelno vrednost) do praznega bloka celo manjša. Zato je implicitno bločno modeliranje še posebej neprimerno za omrežja, kjer nastopajo izstopajoče vrednosti na povezavah.

Ena od možnih rešitev, ki bistveno izboljša dobljena razbitja je, da se tudi pri implicitnem bločnem modeliranju, tako kot pri homogenem, ne uporablja ničelnih blokov. Vendar pa se v tem primeru tudi pri implicitnem bločnem modeliranju pojavijo problem praznega bloka in vse njegove posledice, kot so opisane v prejšnji točki. Vendar pa s takim popravkom implicitno bločno modeliranje postane zelo nenavaden pristop. Lahko ga označimo že za homogeno bločno modeliranje z veliko mero variabilnosti.

*Slika 16.1: Omrežje z razbitjem*

16.4.6 Računanje vrednosti v bločnem modelu

V primeru binarnih omrežij je dovolj, da v bločnem modelu določimo le tipe blokov. V primeru obteženih omrežij pa s tem izgubimo ogromno informacij. Tako je že Batagelj (1997: 148, 151) v primeru obteženih omrežij predlagal računanje vrednosti na povezavah v bločnem modelu. Predlagal dwoje možnih smernic, ki naj bi se uporabljale pri računanju vrednosti na povezavah v bločnem modelu:

1. Vrednost na povezavi, ki predstavlja prazen blok mora biti 0 in vrednost povezave, ki predstavlja blok, kjer so vse vrednosti enake mora biti enaka tem vrednostim

2. Vrednost na povezavi mora biti taka, da je pokrito z 'mnogoternostjo' tipa bloka, dobimo vsoto vseh vrednosti v bloku. Mnogoternost bloka je za popoln blok definiral kot število celic v bloku, za vrstično-regularen kot število stolpcev in za regularen kot maksimum števila stolpcev in števila vrstic.

Predlagal je tudi konkretna pravila za računanje vrednosti na povezavah in sicer v skladu z
prvo smernico za omrežja, kjer so povezave merjene na ordinalni lestvici, in v skladu z drugo smernico za omrežja, kjer so povezave merjene z intervalno lestvico. Menim, da upoštevanje druge smernice poda neustrezne vrednosti, medtem ko se mi zdi prva smernica ustrezn. Zato sem tudi oblikoval svoje predloge za računanj vrednosti v bločen modelu v skladu s to smernico. Oblikoval sem več predlogov, in sicer ločenega za vsako vrsto bločnega modeliranja. Ti predlogi so bili oblikovani tako, da se vrednosti na povezavah računajo na način, ki je v tesni zvezi z načinom računanja odstopanja od idealnih blokov za to vrsto bločnega modeliranja. Na primer, za homogeno bločno modeliranje sem predlagal, da se za vrednost v bločnem modelu uporabi kar tista vrednost, od katere se računajo odkloni pri računanju odstopanj. Problem se pojavi le pri f-regularnem bloku, kjer sta dve tako vrednosti. Kot eno izmed rešitev v tem primeru predlagam računanje srednje vrednosti teh dveh vrednosti.

Pravila, pridobljena na podlagi homogenega bločnega modeliranja z vso kvadratov kot mero variabilnosti, predlagam z manjšim popravkom (za primer (f-)regularnih blokov) tudi kot bolj splošna pravila. Po teh pravilih se pri večini blokov vrednost v bločnem modelu računa kot povprečje vrednosti, ki so pomembne za posamezen tip bloka. To so lahko na primer vse vrednosti v polnem bloku, vrednosti v dominantni vrstici ali stolpcu pri dominantnih blokih, največje vrednosti po vrsticah ali stolpcih v primeru funckijskih blokov, povprečje vrednosti funkcije f po vrsticah v vrstico-f-regularnih blokih, itd. Vrednost praznega bloka pa je v skladu s prvo smernico Batagelja enaka 0.

16.4.7 Predlagane izboljšave, spremembe, alternativne uporabe

V disertaciji prelagam tudi kar nekaj možnih sprememb, idej za izboljšavo ali alternativnih uporab predlaganih pristopov. Večinoma so tu predlagane nepreizkušene ideje. Tukaj so samo našteti in zelo na kratko opisani.

Tako predlagam način, kako je možno na istem omrežju sočasno uporabiti več vrst bločnega modeliranja. To lahko storimo tako, da ustrezno združimo odstopanja, izračunan glede na različne vrste bločnega modeliranja. Glavni problem pri tem pa je ustrezno obteževanje teh odstopanj, saj v izvorni obliki ta odstopanja niso merjena na enako lestvici in jih zato ne smemo seštevati. Za nekatere pare so podani tudi predlogi za način obteževanja.

Drug predlog se nanaša na problem, da pri obteženem bločnem modeliranju odstopanja od 0 niso omejena, od parametra m pa so (na največ m). Problem lahko rešimo, tako da omrežje krmimo pri vrednosti parametra m, se pravi, da vse večje vrednosti rekodiramo v m. Vendar pa v tem primeru tudi izgubimo informacijo o vrednostih nad m.

Predlagam tudi možno izboljšavo za implicitno bločno modeliranje, kadar se uporablja z praznimi bloki. Ugotavljam, je del težav v veliki občutljivosti tega pristopa na izstopajoče vrednosti. V tem smislu predlagam kot možno omilitev težav drugačen način računanja vrednosti, od katere se računajo odstopanja. Kadar to ni 0, je to največja vrednost. Vpliv izstopajočih vrednosti bi lahko omilili, če bi pri računanju največje vrednosti izločili nek odstotek izstopajočih vrednosti. Posledično bi bilo potrebno tudi popraviti način odstopanj in
sicer tako, da bi se upoštevalo, da so lahko vrednosti, za katere se računa odstopanje, sedaj tudi večje od te popravljene največje vrednosti. Glede tega v disertaciji predlagam dve možnosti.

Pišem pa tudi o možnem načinu prilagoditve predlaganih pristopov na označena (obtežena) omrežja, alternativnem načinu računanja odstopanja od f-regularnih blokov pri homogenem bločnem modeliranju, uporabi obteženega bločnega modeliranja na binarnih omrežjih in uporabi posplošenega bločnega modeliranja na vrstično in stolpčno normaliziranih omrežjih.

16.4.8 Implementacija

Vsi pristopi razviti v tej disertaciji, ter tudi nekateri drugi, so bili implementirani v paketu blockmodeling (Žiberna, 2007a), dodatku za statistični program in okolje R (R Development Core Team 2006). Tako so v tem paketu implementirani vse omenjene vrste posplošenega bločnega modeliranja, nove in obstoječe različice algoritma REGE in računanje različnosti ali podobnosti v skladu s strukturno enakovrednostjo.

16.5 Primeri

Razviti pristopi k bločnem modeliranju in še nekateri drugi so preverjeni na štirih empiričnih in dveh umetnih primerih, ki so predstavljeni v devetem poglavju. V večini primerov je poudarek na posplošenem bločnem modeliranju, še posebej na novih pristopih, razvitih za omrežja z vrednostmi na povezavah. Dva primera odstopata od tega pravila. V teh dveh primerih je preučevano delovanje različnih različic algoritma REGE.

Z uporabo izbranih pristopov na prej omenjenih primerih je odkrito ali potrjeno kar nekaj značilnosti teh pristopov oziroma njihovih implementacij. V nadaljevanju predstavljeni rezultati temeljijo predvsem na štirih empiričnih primerih.

Eno prvih vprašanj, s katerim se srečamo, ko želimo z bločnim modeliranjem analizirati neko empirično omrežje, je, kateri pristop naj uporabimo. V idealnih okoliščinah bi izbrali pristop, katerega definicija enakovrednosti oziroma idealnih blokov najbolj ustreza teoretičnemu problemu, ki ga želimo rešiti. Žal pa pogosto problem ni dovolj natančno definiran, še posebej, če je analiza bolj raziskovalne (eksplorativne) narave. V tem primeru nam lahko pomaga tudi spodaj navedene lastnosti, prednosti in slabosti pristopov.

V splošnem lahko trdimo, da so rezultati, pridobljeni s posplošenim bločnim modeliranjem, dobitni. Razbitja enot pridobljena z binarnim bločnim modeliranjem sicer v nekaterih primerih niso najbolj primerna, ali pa je rezultat analize veliko število razbitij (tudi po več 100 razbitij, ki so optimalna glede na vrednost kriterijske funkcije). Kot najprimernejša rešitev je ponavadi izbrano razbitje, pridobljeno z uteženim bločnim modeliranjem, vendar pa je to lahko vsaj deloma tudi posledica dejstva, da je uporabljeno več različnih vrednosti za parameter m, nato pa izbrano najustreznejše razbijte. Vendar pa je potrebno omeniti, da ima trenutno binarno bločno modeliranje edinstveno prednost pred ostalimi pristopi posplošenega bločnega modeliranja. Je namreč edini pristop posplošenega bločnega modeliranja, ki je implementiran
v Pajku (Batagelj in Mrvar, 2006), implementacija tu pa je mnogo (tudi 1000-krat) hitrejša od tiste v paketu blockmodeling.

Homogeno bločno modeliranje nima teh pomanjkljivosti. Njegova glavna pomanjkljivost je tako imenovani ‘problem praznega bloka’. Ta pomanjkljivost ima dve neželeni posledici, ki se kažeta tudi pri uporabi tega pristopa na primerih v devetem poglavju. Prva posledica je, da samo po sebi homogeno bločno modeliranje ne more ločiti med praznim blokom in ostalimi vrstami blokov. Druga je, da homogeno bločno modeliranje pogosto najde ‘prazne’ bloke, vendar pa vseeno vsebujejo več povezav kot prazni bloki, pridobljenimi z drugimi pristopi.

Implicitno bločno modeliranje ponavadi na podlagi razbitja enot, vendar pogosto le, če se uporablja brez praznih blokov. Če se na primer na tak način, se ‘problem praznega bloka’ nanaša tudi na ta pristop, vendar pa imajo bloki, ki so podobni praznim, pri tem pristopu ponavadi vseeno manj močnejših povezav kot taki bloki, pridobljeni x homogenim bločnim modeliranjem. Kadar se implicitno bločno modeliranje uporablja z praznimi bloki, so pridobljena razbitja ponavadi slabša, bločni modeli pa neustrezni.

Rezultati posrednih pristopov so v večini primerov relativno dobri, vendar ponavadi ne tako dobre kot tisti, ki so bili pridobljeni s posplošenim bločnim modeliranjem. V nekaterih primerih so posredni pristopi sposobni identificirati ustrezna razbitja pri nekaterih, ne pa pri vseh izbranih številih skupin. Vendar pa je zaradi hitrosti računanja njihova uporaba zelo priporočljiva, vsaj kot začetni korak v analizi (za oceno primernosti strukturne in regularne enakovrednosti in števila skupin).

Omenil sme že, da je posplošeno bločno modeliranje ponavadi dalo boljše rezultate kot posredni pristopi. Izjema so omrežja, kjer ‘velikost’ enot vpliva na vrednosti na povezavah (na primer pri omrežjih toka ogljika med živimi bitiji). Tu različica algoritma REGE, ki je razvita posebej za taka omrežja, najde dobre rezultate. Posplošeno bločno modeliranje bi morda tudi lahko našlo razbitja, če bi bilo uporabljeno na ustrezno normaliziranih omrežjih (kar pa ni bilo), pri uporabi na vrsti in stolpčno normaliziranih matrikah pa takih razbitij ne najde.

V dveh primerih so izračunane tudi vrednosti na povezavah v bločnem modelu na podlagi večjega števila povezav ob upoštevanju vsote kvadratov in uteženega bločnega modeliranja dajo najustreznejše vrednosti. Pravila na osnovi homogenega bločnega modeliranja na osnovi vsote kvadratov dajajo več poudarka vrednostmi na povezavah (ob upoštevanju vrste blokov), tista na podlagi uteženega bločnega modeliranja pa dajo več poudarka vzorcu povezav v blokih (ob upoštevanju, kako daleč so vrednosti povezav od 0 ali od parametra m).
16.6 Simulacije

V desetem poglavju je predstavljena razmeroma obsežna simulacijska študija, kjer je ovrednoteno delovanje predlaganih pristopov na uteženih, regularnih omrežjih. Simulacije so narejene v dveh fazah. V tem povzetku so predstavljeni predvsem rezultati druge faze. Prva faza je bila namenjena predvsem kot začetna faza, saj so omrežja z le 11 enotami premajhna za zanesljive rezultate na regularnih omrežjih, saj so majhnih regularni bloki (kot posledica majhnih omrežij) razmeroma podobni polnim blokom.

Najbolj presenetljiv rezultat so razmeroma dobri rezultati metod na podlagi strukturne enakovrednosti na omrežjih, generiranih na podlagi maksimum-regularne enakovrednosti. Rezultati homogenačnego bločnega modeliranja na osnovi vsote kvadratov za strukturno enakovrednost (neposredni pristop) so še posebej dobri. Rezultati teh pristopov so sicer slabši kot tisti za regularno enakovrednost, vendar pa precej boljši, kot je bilo moč pričakovati.

Rezultati simulacij kažejo tudi, da so pristopi, razviti v tej disertaciji, posebej homogeno in implicitno bločno modeliranje, boljši pristopi za iskanje regularnih razbitij v regularnih omrežjih z vrednostmi na povezavah kot obstoječi posredni in neposredni pristopi. Algoritmi REGE se dobro obnesejo v situacijah, za katere so razviti, to je v situacijah, kjer je iskano razbitje največje regularno razbitje. Vendar pa rezultati vseeno niso tako dobri, če je iskano razbije največje regularno razbitje šele potem, ko upoštevamo tudi vrednosti na povezavah. V splošnem se v simulacijah najbolje obneseta implicitno in še posebej homogeno (z maksimum-regularnimi bloki) bločno modeliranje.

Binarno in obteženo bločno modeliranje se razmeroma dobro obneseta v prvi fazi, še posebej ko iskano razbitje ni največje regularno razbitje, če upoštevamo tudi vrednosti na povezavah. Žal pa v drugi fazi večinoma odpovesta. Še posebej rezultati za binarno bločno modeliranje kažejo, da je problem vsaj deloma v načinu iskanja ustrezne rešitve, saj ima pravilno razbitje (tisto, ki je uporabljeno pri generiranju omrežij) praviloma manjša odstopanja od idealne rešitve kot tisto razbitje, ki je najdeno z lokalno optimizacijo.

Vendar pa se binarno in obteženo bločno modeliranje slabo obneseta le, kadar se uporabita brez v naprej določenega bločnega modela. Kadar je bločni model v naprej določen, sta ta dva pristopa med najboljšimi pristopi. Uporaba v naprej določenega bločnega modela tudi znatno izboljša rezultate implicitnega bločnega modeliranja v situacijah, kjer iskano razbitje ni največje regularno razbitje.

Pri simulacijah na omrežjih, generiranih na podlagi posplošene enakovrednosti, posplošeno bločno modeliranje nakaže svoje potenciale, pa tudi potrebo po dodatnih raziskavah, še posebej v zvezi s homogenim bločnim modeliranjem.

16.7 Zaključek

V disertaciji so razviti novi pristopi k bločnemu modeliranju, predvsem pristopi za posplošeno bločno modeliranje uteženih omrežij, to je omrežij z vrednostmi na povezavah. Razvite pa so tudi nove verzije algoritma REGE, algoritma za računanje podobnosti in
različnosti v skladu z regularno enakovrednostjo. Poleg tega so v disertaciji pregledani tudi drugi pristopi k bločnem modeliranju ter ostala relevantna teorija. Tu je vrednost še posebej izpostaviti pregled enakovrstnosti, ki se uporabljajo pri bločnem modeliranju. Še posebej se tu osredotočam na regularno enakovrednostjo, kjer kljub razmeroma obsežni literaturi (e.g., White in Reitz; 1983; Borgatti in Everett, 1992b; Everett in Borgatti, 1993; 1994; Batagelj in drugi, 1992a) ostaja še precej nejasnosti, še posebej pri obteženih omrežjih. V disertaciji izpostavim te nejasnosti, oziroma različna razumevanja koncepta, ki se pojavljajo v literaturi. Poleg tega pa tudi definiram novo enakovrednost podobno regularni, ki jo imenujem $f$-regularna enakovrednost.


Vsi razviti in nekateri obstoječi pristopi so bili preizkušeni tudi na primerih in v simulacijah na regularnih omrežjih. Rezultati evalvacije na primerih so pogosto nasprotnoči tistim, ki so pridobljeni v simulacijah. V primerih se je največkrat najbolje obneslo bločno modeliranje, kar pa je lahko tudi posledica tega, da je bilo vedno preizkušeno več različic glede na vrednost paramtera $m$. Binarno bločno modeliranje se je ponavadi dalo le malo slabše rezultate. Homogeno bločno modeliranje je na primerih sicer dalo zadovoljive, a pogosto ne zelo dobre rezultate, najverjetneje kot posledica problema praznega bloka. Implicitno bločno modeliranje je dalo, kadar se je uporabljalo z praznim blokom, neprimerne bločne modele in malce slabša razbitja. Kadar pa se je uporabljalo brez praznega bloka, bločnih modelov sicer ne dobimo (ker smo uporabili le en tip bloka), dobljena razbijta pa so se izboljšala. Posredni pristopi so običajno dali slabše rezultate, predvsem pri nekaterih številnih skupin. Novo razviti algoritmi REGE za 'običajna' omrežja so delovali podobno kot originalni. Zato pa so verzije algoritma REGE za omrežja, kjer so vrednosti na povezavah odvisne od 'velikosti' enot dale na takih omrežjih dobre rezultate.

Na simulacijah so bili rezultati skoraj obratni. Homogeno bločno modeliranje je bil na splošno najboljši pristop. Na primerih pa je občasno obneslo implicitno bločno modeliranje (brez praznega bloka). Rezultati binarnega in obteženega bločnega modeliranja so bili na omrežjih z 25-timi enotami (na tistih z le 11 enotami so bili bistveno boljši) zelo slabi. V simulacijah pa so se pokazale tudi prednosti. Uporaba v naprej določenih bločnih modelov. Z uporabo le teh so vsi pristopi, kjer so bilo uporabljena, zelo dobre in občasno bločno modeliranje ponavadi rezultate, primerljive ali celo boljše od najboljših rezultatov. Kljub temu, da so bila simulirana regularna omrežja, so se presenetljivo dobro obnesli tudi pristopi za strukturno enakovrednost, predvsem homogeno bločno modeliranje za strukturno enakovrednost.

V splošnem lahko trdimo, da so novo razviti pristopi opravičili svoj obstoj, saj so se praviloma na obteženih omrežjih obnesli bolje kot obstoječi. Ljub temu pa ostaja še veliko
možnosti za izboljšave, predvsem pri homogenem in implicitnem bločnem modeliranju. V disertaciji predlagam tudi nekaj možnih načinov izboljšave, ki pa bi jih bilo potrebno še natančneje preučiti in preizkusiti.

Razviti pristopi so primerni tudi za analizo dvovrstnih omrežij, vendar pa na takih omrežjih še niso bili preizkušeni. Poleg tega velja v prihodnosti raziskati tudi možnosti za razširitev teh pristopov na trirazsežna (eno, dvo in trivrstna)omrežja (matrike), večrelacijska omrežja ter poiskati nove vrste blokov. Za uporabnost predstavljenih pristopov pa bi bilo nujno tudi izboljšati njihovo implementacijo, tako da bi bilo za njihovo uporabo potrebno manj časa in da bi bila možna uporaba tudi na malce večjih omrežjih.
Appendix A: Example from Section 7.3.3

The network and the partition in Figure 7.1 were analyzed with all blockmodeling types using structural equivalences. The more detailed results are as follows.

As already mentioned, binary blockmodeling with threshold $t$ equal to 1 or lower and valued blockmodeling with parameter $m$ equal to 1 produced the following image when allowing null and complete block types:

```
1 2
1 "null" "com"
2 "com" "null"
```

On the other hand, implicit blockmodeling when allowing null and complete block types produces the following image:

```
1 2
1 "null" "com"
2 "null" "null"
```

Just for classification reasons when only allowing null and complete block types, it is irrelevant whether block size and maximum normalizations are used. The values of the inconsistencies below are for cases when these two normalizations are not used. The classification of an empirical block into block types is done by comparing block type inconsistencies for allowed block types block by block. Null block inconsistencies are computed as the sum of deviations from 0 for all cells in a block, while complete block inconsistencies are computed as the sum deviations from the maximum of non-null cell values in a block. If there are only nulls in a block, the maximum of all (non-null) values in the matrix (network) is used instead. So for a complete block deviations are computed from the following values in the corresponding blocks:

```
[,] [,2]
[1,] 3 1
[2,] 3 3
```

As we can see, they are different across blocks, which is one of the main characteristics of implicit blockmodeling and what distinguishes it from valued blockmodeling. Although this allows implicit blockmodeling to find complete blocks that differ with respect to the values of cells in them (e.g. one complete block where all values are around 1, one where all cell values are around 3 etc.), it also makes it possible that a block ($R(2, 1)$) with all values equal or

---

132 Suggestion from Subsection 7.3.1 that the maximum of non-null values for the block where all values are 0 should be set to the maximum of the network is used.
greater than some other block \((R(1, 2))\) is classified as null, while the other is classified as complete.

When we compute the deviations from 0 for a null block type and from the above presented values for a complete block type we get the following block type inconsistencies:

**Null block inconsistencies:**

<table>
<thead>
<tr>
<th></th>
<th>[,1]</th>
<th>[,2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>[2,]</td>
<td>11</td>
<td>0</td>
</tr>
</tbody>
</table>

**Complete block inconsistencies:**

<table>
<thead>
<tr>
<th></th>
<th>[,1]</th>
<th>[,2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>[2,]</td>
<td>16</td>
<td>18</td>
</tr>
</tbody>
</table>

For all blocks but \(R(2, 1)\) exactly one of the block type inconsistencies is 0 and the blocks are classified as the block types corresponding to that block type inconsistency. For block \(R(2, 1)\) we can see that the block type inconsistency is lower for a null block and therefore the block \(R(2, 1)\) is classified as null. The values in this block (3 and eleven 1s) are closer to 0 than to the block maximum, which is 3. The relatively large complete block inconsistency is caused by one value that is considerably larger than the others. This large value causes a large maximum and since the deviations are computed from that maximum the deviations for most cell values (values that are relatively low) are large.
Appendix B: Rules for selecting the dominant row of a column in row- or column-dominant blocks for computing tie values in reduced graphs based on homogeneity blockmodeling

<table>
<thead>
<tr>
<th></th>
<th>off-diagonal blocks</th>
<th>diagonal blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>row-dominant</strong></td>
<td><strong>version 1</strong></td>
<td><strong>version 2</strong></td>
</tr>
<tr>
<td></td>
<td>$dr = \underset{i=\text{argmax}}{\text{arg min}} \left{ \text{Cent B}<em>{i,j}\text{Cent B}</em>{k,j} \right} \left( \text{Dev}\left(b_{ij}\right) \right)$</td>
<td>$dr = \underset{i=\text{argmax}}{\text{arg min}} \left{ \text{Cent B}<em>{i,j}\text{Cent B}</em>{k,j} \right} \left( \text{Dev}\left(b_{ij}\right) + \left</td>
</tr>
<tr>
<td></td>
<td>$dr = \underset{i=\text{argmax}}{\text{arg min}} \left{ \text{Dev}\left(b_{ij}\right) \right}$</td>
<td>$dr = \underset{j=\text{argmax}}{\text{arg min}} \left{ \text{Dev}\left(b_{ij}\right) + \left</td>
</tr>
<tr>
<td></td>
<td>$dr = \underset{i=\text{argmax}}{\text{arg min}} \left{ \text{Dev}\left(b_{ij}\right) \right}$</td>
<td>$dr = \underset{j=\text{argmax}}{\text{arg min}} \left{ \text{Dev}\left(b_{ij}\right) + \left</td>
</tr>
<tr>
<td><strong>row-dominant</strong></td>
<td><strong>version 3</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$dc = \underset{j=\text{argmax}}{\text{arg min}} \left{ \text{ss}\left(b_{ij}\right) \right}$</td>
<td></td>
</tr>
<tr>
<td><strong>column-dominant</strong></td>
<td><strong>version 1</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$dc = \underset{j=\text{argmax}}{\text{arg min}} \left{ \text{ss}\left(b_{ij}\right) \right}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$dc = \underset{j=\text{argmax}}{\text{arg min}} \left{ \text{ss}\left(b_{ij}\right) \right}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$dc = \underset{j=\text{argmax}}{\text{arg min}} \left{ \text{ss}\left(b_{ij}\right) \right}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$dc = \underset{j=\text{argmax}}{\text{arg min}} \left{ \text{ss}\left(b_{ij}\right) \right}$</td>
<td></td>
</tr>
</tbody>
</table>
Legend:

- **B** matrix of block $R(C_i, C_j)$
- **B_{i,j]** the i-th row of t matrix B
- **B_{j,i]** the j-th column of matrix B
- **b_{i,j}** an element of matrix B defined by i-th row and j-th column
- **n_r** number of rows in a block = card $C_i$
- **n_c** number of columns in a block = card $C_j$
- **diag(B)** a vector of the diagonal elements of matrix B
- **Cent x** a measure of central tendency – mean for sum of squares blockmodeling, median for absolute deviations blockmodeling
- **Dev x** a measure of deviation – sum of square deviations from the mean for sum of squares blockmodeling, sum of absolute deviations from the median for absolute deviations blockmodeling
- **p** 2 for sum of squares blockmodeling, 1 for absolute deviations blockmodeling
- **dr** dominant row
- **dc** dominant column
Appendix C: Pairwise comparisons of partitions obtained for the ‘Sweet’ network (Example 2) with generalized blockmodeling computed using Adjusted Rand Index

<table>
<thead>
<tr>
<th></th>
<th>bin, t1</th>
<th>bin, t2</th>
<th>bin, t3-10</th>
<th>bin, t5-30</th>
<th>val, max, m1-4</th>
<th>val, max, m5</th>
<th>val, max, m10-40</th>
<th>val, max, m50-90</th>
<th>val, sum, m1</th>
<th>val, sum, m10-50</th>
<th>val, sum, m60-90</th>
<th>ss, max</th>
<th>ss, mean</th>
<th>ad, max</th>
<th>ad, mean</th>
<th>imp, reg only</th>
<th>imp, null&amp;reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin, t1</td>
<td>0.00</td>
<td>-0.07</td>
<td>-0.07</td>
<td>-0.04</td>
<td>0.01</td>
<td>-0.07</td>
<td>-0.10</td>
<td>-0.11</td>
<td>0.01</td>
<td>-0.10</td>
<td>-0.11</td>
<td>-0.12</td>
<td>-0.04</td>
<td>-0.11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bin, t2</td>
<td>-0.07</td>
<td>0.00</td>
<td>0.62</td>
<td>0.46</td>
<td>0.33</td>
<td>0.33</td>
<td>0.46</td>
<td>0.19</td>
<td>0.33</td>
<td>0.46</td>
<td>0.19</td>
<td>0.62</td>
<td>0.46</td>
<td>0.19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bin, t3-10</td>
<td>-0.07</td>
<td>0.62</td>
<td>0.00</td>
<td>0.80</td>
<td>0.62</td>
<td>0.62</td>
<td>0.80</td>
<td>0.19</td>
<td>0.62</td>
<td>0.80</td>
<td>0.19</td>
<td>0.62</td>
<td>0.80</td>
<td>0.19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bin, t5-30</td>
<td>-0.04</td>
<td>0.46</td>
<td>0.80</td>
<td>0.00</td>
<td>0.80</td>
<td>0.80</td>
<td>0.80</td>
<td>0.12</td>
<td>0.80</td>
<td>0.80</td>
<td>0.12</td>
<td>0.80</td>
<td>0.33</td>
<td>0.46</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val, max, m1-4</td>
<td>0.01</td>
<td>0.33</td>
<td>0.62</td>
<td>0.80</td>
<td>0.00</td>
<td>0.62</td>
<td>0.47</td>
<td>0.07</td>
<td>1.00</td>
<td>0.47</td>
<td>0.07</td>
<td>0.47</td>
<td>0.62</td>
<td>0.22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val, max, m5</td>
<td>-0.07</td>
<td>0.33</td>
<td>0.62</td>
<td>0.80</td>
<td>0.62</td>
<td>0.00</td>
<td>0.80</td>
<td>0.02</td>
<td>0.62</td>
<td>0.80</td>
<td>0.02</td>
<td>0.62</td>
<td>0.46</td>
<td>0.62</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val, max, m10-40</td>
<td>-0.10</td>
<td>0.46</td>
<td>0.80</td>
<td>0.62</td>
<td>0.47</td>
<td>0.80</td>
<td>0.00</td>
<td>0.05</td>
<td>0.47</td>
<td>1.00</td>
<td>0.05</td>
<td>1.00</td>
<td>0.80</td>
<td>0.61</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val, max, m50-90</td>
<td>-0.11</td>
<td>0.19</td>
<td>0.19</td>
<td>0.12</td>
<td>0.07</td>
<td>0.02</td>
<td>0.05</td>
<td>0.07</td>
<td>0.05</td>
<td>1.00</td>
<td>0.05</td>
<td>1.00</td>
<td>0.19</td>
<td>0.16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val, sum, m1-5</td>
<td>0.01</td>
<td>0.33</td>
<td>0.62</td>
<td>0.80</td>
<td>1.00</td>
<td>0.62</td>
<td>0.47</td>
<td>0.07</td>
<td>0.00</td>
<td>0.47</td>
<td>0.07</td>
<td>0.47</td>
<td>0.62</td>
<td>0.22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val, sum, m10-50</td>
<td>-0.10</td>
<td>0.46</td>
<td>0.80</td>
<td>0.62</td>
<td>0.47</td>
<td>0.80</td>
<td>1.00</td>
<td>0.05</td>
<td>0.47</td>
<td>0.00</td>
<td>0.05</td>
<td>1.00</td>
<td>0.80</td>
<td>0.61</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val, sum, m60-90</td>
<td>-0.11</td>
<td>0.19</td>
<td>0.19</td>
<td>0.12</td>
<td>0.07</td>
<td>0.02</td>
<td>0.05</td>
<td>1.00</td>
<td>0.07</td>
<td>0.05</td>
<td>0.00</td>
<td>0.19</td>
<td>0.16</td>
<td>0.16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ss, max</td>
<td>-0.10</td>
<td>0.46</td>
<td>0.80</td>
<td>0.62</td>
<td>0.47</td>
<td>0.80</td>
<td>1.00</td>
<td>0.05</td>
<td>0.47</td>
<td>1.00</td>
<td>0.05</td>
<td>0.80</td>
<td>0.62</td>
<td>0.05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ad, max</td>
<td>-0.07</td>
<td>0.62</td>
<td>1.00</td>
<td>0.80</td>
<td>0.62</td>
<td>0.62</td>
<td>0.80</td>
<td>0.19</td>
<td>0.62</td>
<td>0.80</td>
<td>0.19</td>
<td>0.80</td>
<td>0.46</td>
<td>0.62</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ss, mean</td>
<td>-0.13</td>
<td>0.20</td>
<td>0.46</td>
<td>0.33</td>
<td>0.22</td>
<td>0.46</td>
<td>0.61</td>
<td>0.16</td>
<td>0.22</td>
<td>0.61</td>
<td>0.16</td>
<td>0.46</td>
<td>0.00</td>
<td>0.42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ad, mean</td>
<td>-0.12</td>
<td>0.62</td>
<td>0.62</td>
<td>0.46</td>
<td>0.33</td>
<td>0.62</td>
<td>0.80</td>
<td>0.09</td>
<td>0.33</td>
<td>0.80</td>
<td>0.09</td>
<td>0.80</td>
<td>0.62</td>
<td>0.42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imp, reg only</td>
<td>-0.04</td>
<td>0.46</td>
<td>0.80</td>
<td>1.00</td>
<td>0.80</td>
<td>0.80</td>
<td>0.62</td>
<td>0.12</td>
<td>0.80</td>
<td>0.62</td>
<td>0.12</td>
<td>0.80</td>
<td>0.33</td>
<td>0.46</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imp, null&amp;reg</td>
<td>-0.11</td>
<td>0.19</td>
<td>0.19</td>
<td>0.12</td>
<td>0.07</td>
<td>0.02</td>
<td>0.05</td>
<td>1.00</td>
<td>0.07</td>
<td>1.00</td>
<td>0.05</td>
<td>0.19</td>
<td>0.16</td>
<td>0.09</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
bin – binary blockmodeling, val – valued blockmodeling, ss – sum of squares (homogeneity) blockmodeling, ad – absolute deviations (homogeneity) blockmodeling, imp - implicit blockmodeling

- t# - (range of) threshold(s) t for binary blockmodeling,
- m# - (range of) parameter(s) m for valued blockmodeling

max, sum, mean – type of f-regular equivalence (at level m for valued blockmodeling) used

reg only, null&reg – allowed block types (in implicit blockmodeling)
Appendix D: Figure 9 from Luczkovich et al. (2003) – A graph of the St. Marks food web and its image graph

Explanation taken from Luczkovich et al. (2003):

(a) A non-metric-multi-dimensional scaling of all nodes in the St. Marks seagrass ecosystem carbon-flow food web using REGE-derived isotrophic classes from the cluster analysis, and (b) the image graph of the reduced complexity network of the same food web. Colors were used to identify class membership based on the $R^2$ regression analysis (see text and Fig. 8). In the image graph, the codes for blue class \{A\} are \{4,10, 11, 16, 18, 20, 23, 24, 26, 29, 30, 32, 33, 35, 37, 38, 39, 40, 41\} and yellow class \{B\} are \{7, 9, 13, 15, 17, 19, 21, 22, 25, 27, 28, 31, 50\}.

See Appendix E for a complete list of the compartment identification code names.
Appendix E: Appendix B from Luczkovich et al. (2003) – A list of the identification codes and compartment names for the St. Marks seagrass carbonflow food web

Appendix F: Reduced graph produced by Nordlund (2007, 64 – Fig. 2) presented in a similar way as the reduced graph by Luczkovich et al. (2003) presented in Appendix D
Appendix G: Results of simulations in Stage 1 for all settings with values of shape1 parameter 10, 6 and 2.

shape1 = 10

<table>
<thead>
<tr>
<th>Settings</th>
<th>Clear pattern</th>
<th>Not maximal regular</th>
<th>Different dist. par.</th>
<th>Different block max.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Adjusted Rand Index

<table>
<thead>
<tr>
<th>Methods</th>
<th>ss</th>
<th>str</th>
<th>sedist</th>
<th>str</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>bin</th>
<th>reg</th>
<th>halfmax</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>bin</th>
<th>reg</th>
<th>min</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>bin</th>
<th>pre</th>
<th>halfmax</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>bin</th>
<th>pre</th>
<th>min</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

| Methods | val|reg|max|2min |
|---------|----|-----|-----|
| 0.0     | 0.4 | 0.8 |      |

| Methods | val|reg|max|max |
|---------|----|-----|------|
| 0.0     | 0.4 | 0.8 |      |

| Methods | val|pre|max|2min |
|---------|----|-----|-----|
| 0.0     | 0.4 | 0.8 |      |

| Methods | val|pre|max|max |
|---------|----|-----|------|
| 0.0     | 0.4 | 0.8 |      |

<table>
<thead>
<tr>
<th>Methods</th>
<th>ad</th>
<th>reg</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>ss</th>
<th>reg</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>ad</th>
<th>reg</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>ss</th>
<th>reg</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>imp</th>
<th>reg</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

| Methods | imp|wnull|reg|max |
|---------|----|-----|------|
| 0.0     | 0.4 | 0.8 |      |

<table>
<thead>
<tr>
<th>Methods</th>
<th>imp</th>
<th>pre</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGD</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGD.ow</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGE</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGE.ow</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGD</th>
<th>reg</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGD.ow</th>
<th>reg</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGE</th>
<th>reg</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGE.ow</th>
<th>reg</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGD</th>
<th>reg</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGD.ow</th>
<th>reg</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGE</th>
<th>reg</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGE.ow</th>
<th>reg</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGD</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGD.ow</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGE</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGE.ow</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGD</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGD.ow</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGE</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGE.ow</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGD</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGD.ow</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGE</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>REGE.ow</th>
<th>reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>0.8</td>
</tr>
<tr>
<td>Methods</td>
<td>REGD</td>
<td>REGD.ow</td>
</tr>
<tr>
<td>---------------</td>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>Clear pattern</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Not maximal regular</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Different dist. par.</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Different block max.</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Methods**
- ss/str
- sedist/str
- bin/str
- bin/pre/str
- val/str
- ad/str
- imp/str
- REGD/str
- REGD.ow/str
- REGE/str
- REGE.ow/str

**Adjusted Rand Index**

**shape1 = 2**
Appendix H: Results of simulations showing the effect of blocks size normalization in Stage 1 for selected settings with values of \( \text{shape1} \) parameter 10, 6 and 2.

\( \text{shape1} = 10 \)
Appendix I: Results of simulations showing the effect of block size and maximum normalization (and their interaction) on implicit blockmodeling in Stage 1 for selected settings with values of \textit{shape1} parameter 10, 6 and 2.

\[ \text{shape1} = 10 \]
### shape1 = 6

#### Settings

- **Clear pattern**
- **Not maximal regular**
- **Different dist. par.**
- **Different block max.**

#### Methods

- **imp|reg|max|sizeNorm|maxNorm**
- **imp|wnull|reg|max|sizeNorm|maxNorm**
- **imp|pre|max|sizeNorm|maxNorm**
- **imp|reg|max|maxNorm**
- **imp|wnull|reg|max|maxNorm**
- **imp|pre|max|sizeNorm**
- **imp|reg|max|sizeNorm**
- **imp|wnull|reg|max|sizeNorm**
- **imp|pre|max|sizeNorm**
- **imp|reg|max**
- **imp|wnull|reg|max**
- **imp|pre|max**

### shape1 = 2

#### Settings

- **Clear pattern**
- **Not maximal regular**
- **Different dist. par.**
- **Different block max.**

#### Methods

- **imp|reg|max|sizeNorm|maxNorm**
- **imp|wnull|reg|max|sizeNorm|maxNorm**
- **imp|pre|max|sizeNorm**
- **imp|reg|max|maxNorm**
- **imp|wnull|reg|max|maxNorm**
- **imp|pre|max|sizeNorm**
- **imp|reg|max|sizeNorm**
- **imp|wnull|reg|max|sizeNorm**
- **imp|pre|max|sizeNorm**
- **imp|reg|max**
- **imp|wnull|reg|max**
- **imp|pre|max**
Appendix J: Results of simulations in Stage 2 in tabular form with information on the mean of the Adjusted Rand Index, its standard error (se), and the number of repetitions (n) used in its computation

<table>
<thead>
<tr>
<th>General Setting</th>
<th>shape1</th>
<th>regularity not enforced</th>
<th>regularity enforced</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>method</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mean</td>
<td>se</td>
</tr>
<tr>
<td>2</td>
<td>1T</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>REGEO</td>
<td>reg</td>
<td>0.483</td>
</tr>
<tr>
<td></td>
<td>REGEO</td>
<td>reg</td>
<td>0.424</td>
</tr>
<tr>
<td></td>
<td>sedist</td>
<td>str</td>
<td>0.062</td>
</tr>
<tr>
<td></td>
<td>ss</td>
<td>reg</td>
<td>0.049</td>
</tr>
<tr>
<td></td>
<td>ss</td>
<td>reg</td>
<td>0.062</td>
</tr>
<tr>
<td></td>
<td>ss</td>
<td>str</td>
<td>0.022</td>
</tr>
<tr>
<td></td>
<td>val</td>
<td>pre</td>
<td>max</td>
</tr>
<tr>
<td></td>
<td>val</td>
<td>pre</td>
<td>max</td>
</tr>
<tr>
<td></td>
<td>val</td>
<td>reg</td>
<td>max</td>
</tr>
<tr>
<td></td>
<td>val</td>
<td>reg</td>
<td>max</td>
</tr>
<tr>
<td>2</td>
<td>1T</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>REGEO</td>
<td>reg</td>
<td>0.027</td>
</tr>
<tr>
<td></td>
<td>REGEO</td>
<td>reg</td>
<td>0.017</td>
</tr>
<tr>
<td></td>
<td>REGEO</td>
<td>reg</td>
<td>-0.008</td>
</tr>
<tr>
<td></td>
<td>sedist</td>
<td>str</td>
<td>0.011</td>
</tr>
<tr>
<td></td>
<td>ss</td>
<td>reg</td>
<td>max</td>
</tr>
<tr>
<td></td>
<td>ss</td>
<td>reg</td>
<td>mean</td>
</tr>
<tr>
<td></td>
<td>ss</td>
<td>str</td>
<td>0.042</td>
</tr>
<tr>
<td></td>
<td>val</td>
<td>pre</td>
<td>max</td>
</tr>
<tr>
<td></td>
<td>val</td>
<td>pre</td>
<td>max</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mean</td>
<td>SEM</td>
<td>N</td>
</tr>
<tr>
<td>---------------</td>
<td>-------</td>
<td>-------</td>
<td>----</td>
</tr>
<tr>
<td>ad</td>
<td>reg</td>
<td>mean</td>
<td>-0.018</td>
</tr>
<tr>
<td>ad</td>
<td>reg</td>
<td>halfmax</td>
<td>-0.010</td>
</tr>
<tr>
<td>bin</td>
<td>reg</td>
<td>min</td>
<td>0.000</td>
</tr>
<tr>
<td>impreg</td>
<td>max</td>
<td>0.156</td>
<td>0.064</td>
</tr>
<tr>
<td>impwnull</td>
<td>reg</td>
<td>max</td>
<td>0.029</td>
</tr>
<tr>
<td>REGD.ow</td>
<td>reg</td>
<td>0.013</td>
<td>0.018</td>
</tr>
<tr>
<td>REGD</td>
<td>reg</td>
<td>0.015</td>
<td>0.023</td>
</tr>
<tr>
<td>REGE.ow</td>
<td>reg</td>
<td>0.001</td>
<td>0.013</td>
</tr>
<tr>
<td>REGE</td>
<td>reg</td>
<td>-0.002</td>
<td>0.012</td>
</tr>
<tr>
<td>sedist</td>
<td>str</td>
<td>0.187</td>
<td>0.042</td>
</tr>
<tr>
<td>ss</td>
<td>reg</td>
<td>max</td>
<td>0.075</td>
</tr>
<tr>
<td>ss</td>
<td>reg</td>
<td>mean</td>
<td>-0.016</td>
</tr>
<tr>
<td>ss</td>
<td>str</td>
<td>0.394</td>
<td>0.074</td>
</tr>
<tr>
<td>val</td>
<td>reg</td>
<td>max</td>
<td>2min</td>
</tr>
<tr>
<td>val</td>
<td>reg</td>
<td>max</td>
<td>max</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>AR</td>
<td>D</td>
<td>10</td>
</tr>
<tr>
<td>ad</td>
<td>reg</td>
<td>max</td>
<td>0.338</td>
</tr>
<tr>
<td>ad</td>
<td>reg</td>
<td>mean</td>
<td>0.282</td>
</tr>
<tr>
<td>bin</td>
<td>reg</td>
<td>halfmax</td>
<td>0.048</td>
</tr>
<tr>
<td>bin</td>
<td>reg</td>
<td>min</td>
<td>0.027</td>
</tr>
<tr>
<td>impreg</td>
<td>max</td>
<td>0.637</td>
<td>0.057</td>
</tr>
<tr>
<td>impwnull</td>
<td>reg</td>
<td>max</td>
<td>0.174</td>
</tr>
<tr>
<td>REGD.ow</td>
<td>reg</td>
<td>0.540</td>
<td>0.060</td>
</tr>
<tr>
<td>REGD</td>
<td>reg</td>
<td>0.462</td>
<td>0.047</td>
</tr>
<tr>
<td>REGE.ow</td>
<td>reg</td>
<td>0.528</td>
<td>0.046</td>
</tr>
<tr>
<td>REGE</td>
<td>reg</td>
<td>0.440</td>
<td>0.055</td>
</tr>
<tr>
<td>sedist</td>
<td>str</td>
<td>0.006</td>
<td>0.013</td>
</tr>
<tr>
<td>ss</td>
<td>reg</td>
<td>max</td>
<td>0.411</td>
</tr>
<tr>
<td>ss</td>
<td>reg</td>
<td>mean</td>
<td>0.298</td>
</tr>
<tr>
<td>ss</td>
<td>str</td>
<td>0.304</td>
<td>0.061</td>
</tr>
<tr>
<td>val</td>
<td>reg</td>
<td>max</td>
<td>2min</td>
</tr>
<tr>
<td>val</td>
<td>reg</td>
<td>max</td>
<td>max</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>AR</td>
<td>R</td>
<td>10</td>
</tr>
<tr>
<td>ad</td>
<td>reg</td>
<td>max</td>
<td>0.476</td>
</tr>
<tr>
<td>ad</td>
<td>reg</td>
<td>mean</td>
<td>0.451</td>
</tr>
<tr>
<td>REGD/reg</td>
<td>0.514 0.032 64</td>
<td>0.537 0.024 100</td>
<td></td>
</tr>
<tr>
<td>REGE.ow/reg</td>
<td>0.249 0.033 64</td>
<td>0.664 0.017 100</td>
<td></td>
</tr>
<tr>
<td>ss/reg</td>
<td>0.530 0.034 64</td>
<td>0.765 0.033 100</td>
<td></td>
</tr>
<tr>
<td>ss/str</td>
<td>0.071 0.015 64</td>
<td>0.110 0.016 100</td>
<td></td>
</tr>
<tr>
<td>ss/str</td>
<td>0.000 0.006 64</td>
<td>-0.004 0.006 100</td>
<td></td>
</tr>
<tr>
<td>ss/str</td>
<td>0.009 0.007 64</td>
<td>0.105 0.015 100</td>
<td></td>
</tr>
<tr>
<td>val/reg</td>
<td>max</td>
<td>0.135 0.022 64</td>
<td>0.098 0.014 100</td>
</tr>
</tbody>
</table>

| ad/reg|max | 0.092 0.065 20 | 0.359 0.073 64 |
| ad/reg|mean | -0.019 0.006 20 | 0.003 0.010 64 |
| bin/pre/halfmax | 0.013 0.051 20 | 0.212 0.055 64 |
| ss/reg/max | 0.619 0.056 20 | 0.596 0.065 64 |
| ss/reg|mean | 0.285 0.055 20 | 0.301 0.025 64 |
| ss/str | 0.556 0.063 20 | 0.510 0.071 64 |
| val/pre/max| 0.054 0.036 20 | 0.357 0.101 20 |
| imp/reg|max | 0.047 0.043 20 | 0.107 0.058 20 |
| imp/wnull/reg|max | 0.068 0.048 20 | 0.004 0.005 20 |

| REGD.ow/reg | 0.508 0.059 20 | 0.374 0.074 64 |
| REGE.ow/reg | 0.522 0.064 20 | 0.728 0.054 64 |
| sedist/str | 0.171 0.051 20 | 0.212 0.055 64 |
| ss/reg|max | 0.619 0.056 20 | 0.596 0.065 64 |
| ss/reg|mean | 0.285 0.055 20 | 0.301 0.025 64 |
| ss/str | 0.556 0.063 20 | 0.510 0.071 64 |
| val/pre/max| 0.054 0.036 20 | 0.357 0.101 20 |
| imp/reg|max | 0.047 0.043 20 | 0.107 0.058 20 |
| imp/wnull/reg|max | 0.068 0.048 20 | 0.004 0.005 20 |

| REGD/reg | 0.177 0.029 64 | 0.017 0.009 100 |
| REGE.ow/reg | 0.514 0.032 64 | 0.537 0.024 100 |
| ss/reg | 0.530 0.034 64 | 0.765 0.033 100 |
| ss/str | 0.071 0.015 64 | 0.110 0.016 100 |
| ss/str | 0.000 0.006 64 | -0.004 0.006 100 |
| ss/str | 0.009 0.007 64 | 0.105 0.015 100 |
| val/reg|max| 0.135 0.022 64 | 0.098 0.014 100 |

| REGD.ow/reg | 0.508 0.059 20 | 0.374 0.074 64 |
| REGE.ow/reg | 0.522 0.064 20 | 0.728 0.054 64 |
| sedist/str | 0.171 0.051 20 | 0.212 0.055 64 |
| ss/reg|max | 0.619 0.056 20 | 0.596 0.065 64 |
| ss/reg|mean | 0.285 0.055 20 | 0.301 0.025 64 |
| ss/str | 0.556 0.063 20 | 0.510 0.071 64 |
| val/pre/max| 0.054 0.036 20 | 0.357 0.101 20 |
| imp/reg|max | 0.047 0.043 20 | 0.107 0.058 20 |
| imp/wnull/reg|max | 0.068 0.048 20 | 0.004 0.005 20 |

| REGD/reg | 0.177 0.029 64 | 0.017 0.009 100 |
| REGE.ow/reg | 0.514 0.032 64 | 0.537 0.024 100 |
| ss/reg | 0.530 0.034 64 | 0.765 0.033 100 |
| ss/str | 0.071 0.015 64 | 0.110 0.016 100 |
| ss/str | 0.000 0.006 64 | -0.004 0.006 100 |
| ss/str | 0.009 0.007 64 | 0.105 0.015 100 |
| val/reg|max| 0.135 0.022 64 | 0.098 0.014 100 |
|        | ad|reg|max | 0.332 0.064 20 0.549 0.070 20 0.768 0.038 32 0.950 0.013 32 | ad|reg|mean | 0.320 0.097 20 0.442 0.096 20 0.772 0.037 32 0.832 0.027 32 | ad|reg|halfmax | 0.312 0.090 20 0.442 0.096 20 0.772 0.037 32 0.832 0.027 32 | ad|reg|min | 0.300 0.085 20 0.442 0.096 20 0.772 0.037 32 0.832 0.027 32 |...
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin</td>
<td>reg</td>
<td>halfmax</td>
<td>-0.051</td>
</tr>
<tr>
<td>bin</td>
<td>reg</td>
<td>min</td>
<td>0.001</td>
</tr>
<tr>
<td>imp</td>
<td>pre</td>
<td>max</td>
<td>0.191</td>
</tr>
<tr>
<td>imp</td>
<td>reg</td>
<td>max</td>
<td>0.252</td>
</tr>
<tr>
<td>imp</td>
<td>wnull</td>
<td>reg</td>
<td>max</td>
</tr>
<tr>
<td>REGD.</td>
<td>ow</td>
<td>reg</td>
<td>0.058</td>
</tr>
<tr>
<td>REGD</td>
<td>reg</td>
<td>0.081</td>
<td>0.028</td>
</tr>
<tr>
<td>REGE.</td>
<td>ow</td>
<td>reg</td>
<td>0.085</td>
</tr>
<tr>
<td>REGE</td>
<td>reg</td>
<td>0.146</td>
<td>0.053</td>
</tr>
<tr>
<td>imp</td>
<td>pre</td>
<td>max</td>
<td>0.003</td>
</tr>
<tr>
<td>imp</td>
<td>reg</td>
<td>max</td>
<td>0.074</td>
</tr>
<tr>
<td>REGD.</td>
<td>ow</td>
<td>reg</td>
<td>0.058</td>
</tr>
<tr>
<td>REGD</td>
<td>reg</td>
<td>0.081</td>
<td>0.028</td>
</tr>
<tr>
<td>REGE.</td>
<td>ow</td>
<td>reg</td>
<td>0.085</td>
</tr>
<tr>
<td>ad</td>
<td>reg</td>
<td>max</td>
<td>0.252</td>
</tr>
<tr>
<td>ad</td>
<td>reg</td>
<td>mean</td>
<td>0.154</td>
</tr>
<tr>
<td>imp</td>
<td>reg</td>
<td>max</td>
<td>0.074</td>
</tr>
<tr>
<td>imp</td>
<td>wnull</td>
<td>reg</td>
<td>max</td>
</tr>
<tr>
<td>REGD.</td>
<td>ow</td>
<td>reg</td>
<td>0.058</td>
</tr>
<tr>
<td>REGD</td>
<td>reg</td>
<td>0.081</td>
<td>0.028</td>
</tr>
<tr>
<td>REGE.</td>
<td>ow</td>
<td>reg</td>
<td>0.085</td>
</tr>
<tr>
<td>sedist</td>
<td>str</td>
<td>0.154</td>
<td>0.031</td>
</tr>
<tr>
<td>ss</td>
<td>reg</td>
<td>max</td>
<td>0.003</td>
</tr>
<tr>
<td>ss</td>
<td>reg</td>
<td>mean</td>
<td>0.074</td>
</tr>
<tr>
<td>ss</td>
<td>str</td>
<td>0.154</td>
<td>0.031</td>
</tr>
<tr>
<td>val</td>
<td>pre</td>
<td>max</td>
<td>2min</td>
</tr>
<tr>
<td>val</td>
<td>pre</td>
<td>max</td>
<td>max</td>
</tr>
<tr>
<td>val</td>
<td>reg</td>
<td>max</td>
<td>2min</td>
</tr>
<tr>
<td>R]10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CP</td>
<td>R</td>
<td>10</td>
</tr>
<tr>
<td>ad</td>
<td>reg</td>
<td>max</td>
<td>0.148</td>
</tr>
<tr>
<td>ad</td>
<td>reg</td>
<td>mean</td>
<td>0.064</td>
</tr>
<tr>
<td>bin</td>
<td>pre</td>
<td>halfmax</td>
<td>0.119</td>
</tr>
<tr>
<td>bin</td>
<td>pre</td>
<td>min</td>
<td>0.038</td>
</tr>
<tr>
<td>bin</td>
<td>pre</td>
<td>halfmax</td>
<td>-0.016</td>
</tr>
<tr>
<td>bin</td>
<td>pre</td>
<td>min</td>
<td>-0.003</td>
</tr>
<tr>
<td>imp</td>
<td>pre</td>
<td>max</td>
<td>0.264</td>
</tr>
<tr>
<td>imp</td>
<td>reg</td>
<td>max</td>
<td>0.391</td>
</tr>
<tr>
<td>imp</td>
<td>reg</td>
<td>max</td>
<td>0.083</td>
</tr>
<tr>
<td>REGD.</td>
<td>ow</td>
<td>reg</td>
<td>0.239</td>
</tr>
<tr>
<td>REGD</td>
<td>reg</td>
<td>0.216</td>
<td>0.061</td>
</tr>
<tr>
<td>REGE.</td>
<td>ow</td>
<td>reg</td>
<td>0.240</td>
</tr>
<tr>
<td>REGE</td>
<td>reg</td>
<td>0.179</td>
<td>0.051</td>
</tr>
<tr>
<td>ss</td>
<td>reg</td>
<td>max</td>
<td>0.094</td>
</tr>
<tr>
<td>ss</td>
<td>str</td>
<td>0.184</td>
<td>0.046</td>
</tr>
<tr>
<td>val</td>
<td>pre</td>
<td>max</td>
<td>2min</td>
</tr>
<tr>
<td>val</td>
<td>pre</td>
<td>max</td>
<td>max</td>
</tr>
<tr>
<td>val</td>
<td>reg</td>
<td>max</td>
<td>2min</td>
</tr>
<tr>
<td>val</td>
<td>reg</td>
<td>max</td>
<td>max</td>
</tr>
</tbody>
</table>

- The table contains various parameters and their corresponding values, including terms like bin, imp, REGD, REGE, sedist, ss, val, ad, and other abbreviations.
- The values are presented in a tabular format, showing different combinations of parameters with their respective numerical values.
- The table appears to be a detailed analysis or summary of data, possibly related to a study or experiment.
<table>
<thead>
<tr>
<th>Column</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
<th>Value 5</th>
<th>Value 6</th>
<th>Value 7</th>
<th>Value 8</th>
<th>Value 9</th>
<th>Value 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>ad/reg/max</td>
<td>0.005</td>
<td>0.161</td>
<td>-0.003</td>
<td>0.014</td>
<td>0.038</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ad/reg/mean</td>
<td>-0.026</td>
<td>-0.038</td>
<td>-0.030</td>
<td>0.000</td>
<td>0.039</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bin/pre/halfmax</td>
<td>0.109</td>
<td>0.046</td>
<td>0.314</td>
<td>0.635</td>
<td>0.056</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bin/pre/min</td>
<td>0.016</td>
<td>0.139</td>
<td>0.093</td>
<td>0.589</td>
<td>0.062</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bin/reg/halfmax</td>
<td>-0.020</td>
<td>0.014</td>
<td>-0.013</td>
<td>0.106</td>
<td>0.041</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bin/reg/min</td>
<td>0.001</td>
<td>0.011</td>
<td>0.007</td>
<td>0.996</td>
<td>0.040</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imp/pre/max</td>
<td>-0.002</td>
<td>-0.015</td>
<td>0.273</td>
<td>0.473</td>
<td>0.075</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imp/reg/max</td>
<td>-0.004</td>
<td>0.051</td>
<td>0.066</td>
<td>-0.017</td>
<td>0.003</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imp/wnull/reg/max</td>
<td>0.025</td>
<td>-0.011</td>
<td>-0.013</td>
<td>0.013</td>
<td>0.015</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGD.ow/reg</td>
<td>-0.002</td>
<td>-0.007</td>
<td>0.011</td>
<td>0.020</td>
<td>0.016</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGD/reg</td>
<td>0.013</td>
<td>-0.012</td>
<td>0.030</td>
<td>0.962</td>
<td>0.021</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGE.ow/reg</td>
<td>0.008</td>
<td>-0.006</td>
<td>0.020</td>
<td>-0.010</td>
<td>0.008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGD/str</td>
<td>-0.035</td>
<td>0.273</td>
<td>0.027</td>
<td>0.046</td>
<td>0.005</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGD/ow</td>
<td>0.160</td>
<td>0.328</td>
<td>0.356</td>
<td>0.905</td>
<td>0.035</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGD/str</td>
<td>0.169</td>
<td>0.226</td>
<td>0.182</td>
<td>0.395</td>
<td>0.048</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GM/reg/max/2min</td>
<td>0.020</td>
<td>0.173</td>
<td>0.063</td>
<td>0.762</td>
<td>0.052</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GM/str</td>
<td>0.125</td>
<td>0.104</td>
<td>0.707</td>
<td>0.755</td>
<td>0.056</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GM/str</td>
<td>-0.008</td>
<td>0.041</td>
<td>0.013</td>
<td>0.023</td>
<td>0.013</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val/reg/str</td>
<td>0.310</td>
<td>0.151</td>
<td>0.021</td>
<td>0.011</td>
<td>0.013</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val/str</td>
<td>0.283</td>
<td>0.614</td>
<td>0.455</td>
<td>0.946</td>
<td>0.015</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val/reg/str</td>
<td>0.145</td>
<td>0.219</td>
<td>0.207</td>
<td>0.307</td>
<td>0.027</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3| C| 1|10

<table>
<thead>
<tr>
<th>Column</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
<th>Value 5</th>
<th>Value 6</th>
<th>Value 7</th>
<th>Value 8</th>
<th>Value 9</th>
<th>Value 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>ad/reg/max</td>
<td>0.257</td>
<td>0.456</td>
<td>0.410</td>
<td>0.738</td>
<td>0.073</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ad/reg/mean</td>
<td>-0.019</td>
<td>-0.018</td>
<td>0.000</td>
<td>0.037</td>
<td>0.006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bin/reg/halfmax</td>
<td>-0.020</td>
<td>0.031</td>
<td>0.000</td>
<td>0.960</td>
<td>0.029</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bin/reg/min</td>
<td>-0.005</td>
<td>0.013</td>
<td>0.016</td>
<td>0.016</td>
<td>0.022</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imp/reg/max</td>
<td>0.251</td>
<td>0.273</td>
<td>0.309</td>
<td>0.538</td>
<td>0.081</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imp/wnull/reg/max</td>
<td>0.024</td>
<td>0.087</td>
<td>0.231</td>
<td>0.226</td>
<td>0.041</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGD.ow/reg</td>
<td>0.250</td>
<td>0.134</td>
<td>0.454</td>
<td>0.758</td>
<td>0.044</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGD/reg</td>
<td>0.218</td>
<td>0.141</td>
<td>0.361</td>
<td>0.783</td>
<td>0.046</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGE.ow/reg</td>
<td>0.233</td>
<td>0.123</td>
<td>0.436</td>
<td>0.656</td>
<td>0.046</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGD/str</td>
<td>0.214</td>
<td>0.140</td>
<td>0.315</td>
<td>0.660</td>
<td>0.042</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGD/ow</td>
<td>0.169</td>
<td>0.226</td>
<td>0.182</td>
<td>0.395</td>
<td>0.048</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGD/str</td>
<td>0.297</td>
<td>0.584</td>
<td>0.445</td>
<td>0.697</td>
<td>0.079</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GM/reg/max/2min</td>
<td>0.020</td>
<td>0.044</td>
<td>-0.029</td>
<td>-0.056</td>
<td>0.003</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GM/str</td>
<td>0.180</td>
<td>0.233</td>
<td>0.352</td>
<td>0.830</td>
<td>0.046</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GM/str</td>
<td>-0.004</td>
<td>0.030</td>
<td>-0.004</td>
<td>0.128</td>
<td>0.052</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val/reg/str</td>
<td>0.158</td>
<td>0.024</td>
<td>0.347</td>
<td>0.152</td>
<td>0.033</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val/str</td>
<td>0.283</td>
<td>0.588</td>
<td>0.458</td>
<td>0.947</td>
<td>0.016</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3| [C]| O4|10

<table>
<thead>
<tr>
<th>Column</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
<th>Value 5</th>
<th>Value 6</th>
<th>Value 7</th>
<th>Value 8</th>
<th>Value 9</th>
<th>Value 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>ad/reg/max</td>
<td>0.257</td>
<td>0.558</td>
<td>0.363</td>
<td>0.880</td>
<td>0.020</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ad/reg/mean</td>
<td>0.021</td>
<td>0.154</td>
<td>0.036</td>
<td>0.321</td>
<td>0.033</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bin/pre/halfmax</td>
<td>0.123</td>
<td>0.445</td>
<td>0.161</td>
<td>0.866</td>
<td>0.029</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bin/pre/min</td>
<td>0.307</td>
<td>0.271</td>
<td>0.316</td>
<td>0.653</td>
<td>0.066</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bin/reg/halfmax</td>
<td>0.042</td>
<td>0.013</td>
<td>0.112</td>
<td>0.001</td>
<td>0.006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bin/reg/min</td>
<td>-0.011</td>
<td>0.020</td>
<td>0.013</td>
<td>0.045</td>
<td>0.020</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imp/pre/max</td>
<td>0.014</td>
<td>0.144</td>
<td>0.034</td>
<td>0.750</td>
<td>0.072</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imp/reg/max</td>
<td>0.163</td>
<td>0.462</td>
<td>0.327</td>
<td>0.717</td>
<td>0.034</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imp/wnull/reg/max</td>
<td>0.006</td>
<td>0.055</td>
<td>0.109</td>
<td>0.079</td>
<td>0.024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGD.ow/reg</td>
<td>0.307</td>
<td>0.588</td>
<td>0.458</td>
<td>0.947</td>
<td>0.016</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGD/reg</td>
<td>0.278</td>
<td>0.589</td>
<td>0.385</td>
<td>0.956</td>
<td>0.014</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGE.ow/reg</td>
<td>0.285</td>
<td>0.566</td>
<td>0.416</td>
<td>0.935</td>
<td>0.018</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGE/reg</td>
<td>0.301</td>
<td>0.572</td>
<td>0.367</td>
<td>0.927</td>
<td>0.018</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sedist/str</td>
<td>0.075</td>
<td>0.151</td>
<td>0.104</td>
<td>0.182</td>
<td>0.019</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ss/reg/max</td>
<td>0.283</td>
<td>0.614</td>
<td>0.455</td>
<td>0.946</td>
<td>0.015</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ss/reg/mean</td>
<td>0.012</td>
<td>0.066</td>
<td>0.038</td>
<td>0.203</td>
<td>0.042</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ss/str</td>
<td>0.145</td>
<td>0.219</td>
<td>0.207</td>
<td>0.307</td>
<td>0.027</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>2min</td>
<td>0.052</td>
<td>0.032</td>
<td>20</td>
<td>0.435</td>
<td>0.043</td>
<td>20</td>
<td>0.056</td>
<td>0.028</td>
<td>20</td>
</tr>
<tr>
<td>-------------------</td>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>----</td>
<td>-------</td>
<td>-------</td>
<td>----</td>
<td>-------</td>
<td>-------</td>
<td>----</td>
</tr>
<tr>
<td>val/pre/max/2min</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val/pre/max/max</td>
<td>0.442</td>
<td>0.045</td>
<td>20</td>
<td>0.475</td>
<td>0.043</td>
<td>20</td>
<td>0.618</td>
<td>0.032</td>
<td>20</td>
<td>0.861</td>
</tr>
<tr>
<td>val/reg/max/2min</td>
<td>-0.012</td>
<td>0.007</td>
<td>20</td>
<td>0.029</td>
<td>0.018</td>
<td>20</td>
<td>0.005</td>
<td>0.009</td>
<td>26</td>
<td>0.274</td>
</tr>
<tr>
<td>val/reg/max/max</td>
<td>0.003</td>
<td>0.021</td>
<td>20</td>
<td>0.005</td>
<td>0.010</td>
<td>20</td>
<td>0.306</td>
<td>0.062</td>
<td>26</td>
<td>0.005</td>
</tr>
<tr>
<td>3</td>
<td>G3</td>
<td>G1</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ad/reg/max</td>
<td>0.099</td>
<td>0.024</td>
<td>20</td>
<td>0.322</td>
<td>0.045</td>
<td>20</td>
<td>0.133</td>
<td>0.024</td>
<td>26</td>
<td>0.643</td>
</tr>
<tr>
<td>ad/reg/mean</td>
<td>0.001</td>
<td>0.013</td>
<td>20</td>
<td>0.050</td>
<td>0.028</td>
<td>20</td>
<td>0.006</td>
<td>0.009</td>
<td>26</td>
<td>0.128</td>
</tr>
<tr>
<td>bin/reg/halfmax</td>
<td>-0.017</td>
<td>0.007</td>
<td>20</td>
<td>0.140</td>
<td>0.034</td>
<td>20</td>
<td>-0.031</td>
<td>0.003</td>
<td>26</td>
<td>0.156</td>
</tr>
<tr>
<td>bin/reg/min</td>
<td>0.014</td>
<td>0.010</td>
<td>20</td>
<td>0.068</td>
<td>0.035</td>
<td>20</td>
<td>-0.011</td>
<td>0.005</td>
<td>26</td>
<td>0.111</td>
</tr>
<tr>
<td>imp/reg/max</td>
<td>0.058</td>
<td>0.022</td>
<td>20</td>
<td>0.430</td>
<td>0.045</td>
<td>20</td>
<td>0.160</td>
<td>0.019</td>
<td>26</td>
<td>0.578</td>
</tr>
<tr>
<td>imp/wnull/reg/max</td>
<td>0.060</td>
<td>0.015</td>
<td>20</td>
<td>0.101</td>
<td>0.040</td>
<td>20</td>
<td>0.071</td>
<td>0.024</td>
<td>26</td>
<td>0.322</td>
</tr>
<tr>
<td>REGD.ow/reg</td>
<td>0.149</td>
<td>0.020</td>
<td>20</td>
<td>0.116</td>
<td>0.033</td>
<td>20</td>
<td>0.207</td>
<td>0.024</td>
<td>26</td>
<td>0.282</td>
</tr>
<tr>
<td>REGD/reg</td>
<td>0.072</td>
<td>0.018</td>
<td>20</td>
<td>0.115</td>
<td>0.033</td>
<td>20</td>
<td>0.123</td>
<td>0.025</td>
<td>26</td>
<td>0.242</td>
</tr>
<tr>
<td>REG.ow/reg</td>
<td>0.117</td>
<td>0.026</td>
<td>20</td>
<td>0.122</td>
<td>0.032</td>
<td>20</td>
<td>0.181</td>
<td>0.022</td>
<td>26</td>
<td>0.257</td>
</tr>
<tr>
<td>REG/reg</td>
<td>0.097</td>
<td>0.021</td>
<td>20</td>
<td>0.175</td>
<td>0.033</td>
<td>20</td>
<td>0.125</td>
<td>0.024</td>
<td>26</td>
<td>0.264</td>
</tr>
<tr>
<td>sedist/str</td>
<td>0.113</td>
<td>0.026</td>
<td>20</td>
<td>0.246</td>
<td>0.029</td>
<td>20</td>
<td>0.126</td>
<td>0.023</td>
<td>26</td>
<td>0.411</td>
</tr>
<tr>
<td>ss/reg/max</td>
<td>0.099</td>
<td>0.023</td>
<td>20</td>
<td>0.358</td>
<td>0.048</td>
<td>20</td>
<td>0.167</td>
<td>0.023</td>
<td>26</td>
<td>0.702</td>
</tr>
<tr>
<td>ss/reg/mean</td>
<td>-0.002</td>
<td>0.016</td>
<td>20</td>
<td>-0.001</td>
<td>0.017</td>
<td>20</td>
<td>0.018</td>
<td>0.011</td>
<td>26</td>
<td>0.056</td>
</tr>
<tr>
<td>ss/str</td>
<td>0.151</td>
<td>0.039</td>
<td>20</td>
<td>0.353</td>
<td>0.035</td>
<td>20</td>
<td>0.247</td>
<td>0.029</td>
<td>26</td>
<td>0.575</td>
</tr>
<tr>
<td>val/reg/max/2min</td>
<td>0.014</td>
<td>0.009</td>
<td>20</td>
<td>0.082</td>
<td>0.028</td>
<td>20</td>
<td>-0.009</td>
<td>0.005</td>
<td>26</td>
<td>0.275</td>
</tr>
<tr>
<td>val/reg/max/max</td>
<td>0.084</td>
<td>0.023</td>
<td>20</td>
<td>0.062</td>
<td>0.022</td>
<td>20</td>
<td>0.200</td>
<td>0.026</td>
<td>26</td>
<td>0.321</td>
</tr>
</tbody>
</table>