Metodološki zvezki, Vol. 6, No. 2, 2009, 99-134

Evaluation of Direct and Indirect Blockmodeling of Regular Equivalence in Valued Networks by Simulations

Aleš Žiberna[1]

Abstract

The aim of the paper is to compare and evaluate different approaches to detecting regular equivalence classes of valued networks. The evaluated approaches include different versions of REGE (indirect approaches) and generalized blockmodeling approaches (direct approaches). In addition to the approaches designed to detect regular equivalence, some approaches designed to detect structural equivalence are also included for comparison.

The evaluation is done by means of simulations. Networks of 11 and 25 units were generated based on different known (*max*-)regular blockmodels and partitions. The obtained partitions were compared to the original (known) partition using the Adjusted Rand Index.

The results show that homogeneity blockmodeling, implicit blockmodeling and REGE are usually the best approaches for detecting regular equivalence. The most surprising result is that methods for detecting structural equivalence preformed relatively well on networks generated based on (*max*)-regular equivalence, better then several approaches designed to detect regular equivalence.

1 Introduction

The aim of this paper is to evaluate different approaches used for finding regular equivalence (White and Reitz, 1983) classes in valued networks on artificially generated valued networks. The evaluated approaches are direct (generalized) blockmodeling (Batagelj, Doreian and Ferligoj, 1992; Doreian, Batagelj and Ferligoj, 2005; Žiberna, 2007a, 2008) and some versions of REGE (White, 1985a; 1985b; Žiberna, 2008).

Some of these approaches, especially older ones, have been already applied to several empirical datasets (Smith and White, 1992; Luczkovich et al., 2003; Nordlund, 2007; Doreian, Batagelj and Ferligoj, 2005), while the newer ones (Žiberna, 2007a, 2008) are still relatively untested. In this paper all these

[1] Faculty of Social Sciences, University of Ljubljana, Kardeljeva pl. 5, SI-1000 Ljubljana; ales.ziberna@fdv.uni-lj.si

approaches are applied to valued networks that are generated according to (*max*-) regular equivalence (Žiberna, 2008) with some random error. In order to get more reliable results the whole procedure (generating networks, applying methods, etc.) was repeated at least 20 times. The aim of this work is to evaluate these approaches and to identify which of these approaches is the most appropriate for detecting (*max*-)regular equivalence classes? The simulations themselves were split into two stages. In Stage 1 (the preliminary stage), the approaches were tested on a larger number of settings on networks with only 11 units (or vertices). In Stage 2, the number of units was increased to 25, while the number of different settings was reduced based on the results of Stage 1. In most cases, only the results of Stage 2 are reported.

Although the approaches that fall into the framework of generalized blockmodeling (Batagelj, Doreian and Ferligoj, 1992; Doreian, Batagelj and Ferligoj, 2005; Žiberna, 2007a, 2008) can be applied also to a number of other problems, finding regular equivalence classes is probably one of more complex tasks. Thus the evaluation of these approaches in the context of (*max*-)regular equivalence can be also used to:

1. at least partly evaluate these (generalized blockmodeling) approaches in more general setting of blockmodeling of valued networks,
2. determine whether the newer approached for generalized blockmodeling of valued networks are indeed superior to generalized blockmodeling for binary networks when applied to valued networks,
3. and assess alternative criterion functions (normalizations of block inconsistencies) generalized blockmodeling.

In the next section the definition of (*max*-)regular equivalence for valued networks that is used in the generation of simulated networks is presented (2). In Section 3 the methods that are compared in simulations are described. The design of simulations and the settings used in them) are presented in the following section (4). The results of the simulations are presented in Section 5 and their limitations are outlined in Section 6. At the end of the paper we discuss the obtained results (7) and their implications and present ideas for future work (8).

2 (*max*)-regular equivalence

Regular equivalence was first defined by White and Reitz (1983): "Regularly equivalent points (units) are connected in the same way to matching equivalents". However this and also the more formal definition given by White and Reitz (1983) and also later by Batagelj, Doreian and Ferligoj (1992) were suitable only for binary networks and, as noted by Žiberna (2008), were understood or interpreted differently even on binary networks. Borgatti and Everett (1992) provided a formal definition of regular equivalence for valued networks. Algorithms

detecting regular equivalence in valued networks are also all based on some definition of regular equivalence or imply a certain definition of regular equivalence for valued networks, so definitions of regular equivalence can also be obtained from the algorithms REGGE and REGDI (White, 1985a; 1985b) and from implicit approach to blockmodeling of valued networks suggested by Batagelj and Ferligoj (2000). This was done in Žiberna (2008) where all these definitions are also given. Based on these definitions, Žiberna (2008) presented the definition of *f*-regular equivalence as follows:

> Suppose that we have a network N = (U, R[2]) and ≡ is an equivalence relation on U that induces (or corresponds to) a partition **C** then ≡ is an *f-regular equivalence* (where *f* is a selected function, like *sum*, *maximum*, *mean*, etc.) if and only if for all $a, b \in U$ and all $X \in \mathbf{C}$, $a \equiv b$ implies $\underset{i \in X}{f}(r_{ai}) = \underset{i \in X}{f}(r_{bi})$ and $\underset{i \in X}{f}(r_{ia}) = \underset{i \in X}{f}(r_{ib})$.

When the function *f* in this definition is maximum, the equivalence is called a *max*-regular equivalence and is identical to the definition implied by the approach suggested by Batagelj and Ferligoj (2000). This is also the definition used in this paper. Basically, this definition says that a block perfectly complies with the definition of *max*-regular equivalence, if all row maximums and all column maximums are equal. If all blocks induced by a partition that corresponds to a given equivalence satisfy this condition, that the equivalence is a *max*-regular equivalence.

3 Evaluated methods

In this paper two groups of approaches that are capable of detecting regular equivalence classes are evaluated. The approaches from the first group are part of the generalized blockmodeling (Doreian, Batagelj, and Ferligoj, 2005) approach, while the second group consists of different versions of REGE (White, 1985a; 1985b). All the approaches evaluated are described in Žiberna (2008) where they are also applied to empirical networks.

3.1 Generalized blockmodeling

Generalized blockmodeling was first introduced by Doreian, Batagelj and Ferligoj (1994), where the notion of generalized equivalence and several block types were presented. The approach evolved from a direct approach to blockmodeling[3], which was already before capable of detecting regular equivalence clusters (Batagelj,

[2] The relation R is represented by matrix $\mathrm{R} = [r_{ij}]_{\mathrm{n \times n}}$.

[3] Generalized blockmodeling still is a direct approach to blockmodeling.

Doreian and Ferligoj, 1992)[4]. Direct approaches directly search for a partition that best fits the selected equivalence as measured by a criterion function (Batagelj, Ferligoj and Doreian, 1992). The approach is described extensively by Doreian, Batagelj and Ferligoj (2005). In generalized blockmodeling, the equivalence is defined by the allowed block types and possibly their position.

However, it was still applicable only to binary and signed networks, although Batagelj and Ferligoj (2000) have presented some ideas for extending this approach to valued networks. Generalized blockmodeling for binary networks (Doreian, Batagelj, and Ferligoj, 2005) is from now on referred to as *binary blockmodeling*. Žiberna (2007a) presented valued and homogeneity blockmodeling, two types of generalized blockmodeling suited to valued networks. Later he also developed implicit blockmodeling Žiberna (2007b; 2008) based on ideas presented by Batagelj and Ferligoj (2000). All these approaches to generalized blockmodeling of valued networks are capable of detecting regular equivalence classes in valued networks. The approaches are very briefly described below. Longer descriptions can be found in the works cited above.

We can say that binary blockmodeling treats ties either as relevant or as nonexistent (or irrelevant). The inconsistencies are in principle computed as the number of times the tie is present where it is not assumed or vice versa. In the case of regular equivalence, the allowed block types are regular and null. In an ideal null blocks all values are 0, while in an ideal regular blocks there is at least one 1 in each row and each column. As it is here applied to valued networks, these have to be first binarized[5] using an appropriate threshold.

Valued blockmodeling is similar, yet if the tie is not nonexistent or fully relevant, it assesses if a tie is closer to being relevant or closer to nonexistent. The inconsistencies are in principle computed as the sum of differences between the actual tie values and the assumed tie values (0 or the threshold that specifies when a tie is considered relevant). While both of these approaches use the same definition of regular equivalence that is used in this paper in the case of binary networks, this is no longer completely true for valued networks.

Homogeneity blockmodeling 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. This threshold must also be the same for all blocks. 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). In homogeneity blockmodeling tie values (or some statistics computed on them) are assumed to be homogenous (equal) within blocks.

[4] For the purpose of this paper the approach presented in Batagelj, Doreian, and Ferligoj (1992) is sufficient and other advances described in Doreian, Batagelj, and Ferligoj (1994) and Doreian et al. (2005) are not necessary.

[5] Converted to binary networks.

In the case of *f*-regular blocks, values of function *f* computed over rows/columns are assumed to be homogeneous within rows and within columns. The inconsistency of a block is some measure of variability computed on these values. Depending on the measure of variability, two homogeneity blockmodeling approaches are used. These are sum of squares[6] homogeneity blockmodeling and absolute deviations[7].

Implicit blockmodeling is in computational terms very similar to valued blockmodeling. The only differences are that the parameter specifying when a tie is relevant is replaced by block maximum and that initially additional normalization was used[8]. However, in its functioning, it is very similar to homogeneity blockmodeling. The advantage of both homogeneity and implicit blockmodeling is that they (can[9]) use the same definition of regular equivalence as used in this paper (*max*-regular equivalence).

In addition to methods for regular equivalence, one method from the framework of generalized blockmodeling designed for structural equivalence was also used to detect regular equivalence. The method used falls into the homogeneity generalized blockmodeling approach. More precisely, sum of squares homogeneity blockmodeling with only complete blocks are used. The ideal complete block within sum of squares blockmodeling is such that all values in the blocks have the same value. The inconsistency of an empirical block to the ideal block is computed as the sum of squared deviations of the values in the block from the mean of these values.

Below the notation used to identify the above described methods in the results is described:

1. Generalized blockmodeling approach:
   - **bin** - Binary blockmodeling
   - **val** - Valued blockmodeling
   - **ss** – Sum of squares (homogeneity) blockmodeling
   - **ad** – Absolute deviations (homogeneity) blockmodeling
   - **imp** - Implicit blockmodeling
2. Equivalence:
   - **str** – structural equivalence (only used with sum of squares blockmodeling)
   - **reg** – (*f*-)regular equivalence
   - **wnull|reg** – (*f*-)regular equivalence with null blocks (only used in implicit blockmodeling) – indicate that null blocks are used when searching for regular equivalence using implicit blockmodeling, although Žiberna (2008) suggests that they should not be used.

[6] The measure of variation used is the sum of squared deviations from the mean.

[7] The measure of variation used is the sum of absolute deviations from the median.

[8] In the ideas presented by Batagelj and Ferligoj, additional normalizations were used. However, as argued by Žiberna (2008), they are not necessary.

[9] Homogeneity blockmodeling uses the definition of *f*-regular equivalence with any function *f*.

   - **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.
3. Values of the threshold *t* used for binarization (binary blockmodeling)/the parameter *m* (valued blockmodeling):
   - **halfmax/max** – (half[10]) of the empirical maximum of tie values
   - **min/2min** – (twice[11]) the second (the first one is usually 0) minimum of the empirical distribution of all tie values
4. Function *f* used in *f*-regular blocks (not used with structural equivalence or binary blockmodeling): **max** or **mean** (only used in homogeneity blockmodeling)
5. Search procedures:

   Local optimization with 20 random starting partitions – used unless stated otherwise (this and the next option are the default ones and are therefore not marked)
   - Full search – checking all possible partitions (only used in Stage 1 for two-cluster settings).
   - **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[12], as the Adjusted Rand Index is 1 by default) for the correct partition

The 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 is referred to as "val|reg|max|max".

All the evaluated methods are implemented in the `blockmodeling 0.1.2` package (Žiberna, 2006) for R statistical environment (R Development Core Team, 2006) and this implementation was used for all methods. The limitations that the use of this still experimental software presents for the results obtained in this paper are described in Section 6.

[10] Binary blockmodeling

[11] Valued blockmodeling

[12] 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)

3.2 Indirect methods

Indirect approach means that first (dis)similarities compatible with the selected equivalence are computed. Then, some clustering method is used on these (dis)similarities to obtain a partition.

REGE is an algorithm for computing (dis)similarity in terms of regular equivalence. There exists no "closed form[13]" measure of dissimilarity or similarity compatible with regular equivalence. However, White (1985a, 1985b) developed two iterative algorithms, REGGE and REGDI, for computing dissimilarities or similarities (depending on the version) in terms of regular equivalence. As it is explained in Žiberna (2008), these algorithms do not use the same definition of regular equivalence (that is *max*-regular) equivalence as it is used in this paper. Based on them Žiberna (2008) developed versions of these two algorithms that either partly reduce this difference in definitions used (REGDI-OW, a modified version of REGDI) or use exactly *max*-regular equivalence (REGGE-OW, a modified version of REGGE). Only the versions of REGE designed for interval valued networks were used. Therefore, the versions suggested by Borgatti and Everett (1993) were not used, as they were designed for nominal valued networks. In this paper the term REGE is used for all versions of the algorithm discussed above.

REGE evaluates the (degree of) equivalence of units *a* and *b* by trying to match every link (taking values into account) of unit *a* by the most similar link of unit *b* to an equivalent / the most equivalent unit and vice versa (every link of unit *b* an equivalent / the most equivalent link of unit *a*). The difference between the original versions of the algorithm developed by White and the modified versions developed by Žiberna is that in REGGE and REGDI (White, 1985a; 1985b) a link refers to a pair of arcs – an incoming and an outgoing arc, while in REGGE-OW and REGDI-OW (Žiberna, 2008) it only refers to one arc.

If the best match (link) does not have equal (or equal or greater, depending on the version of REGE) values of ties, or if the other points (ends) of ties (from and to *a*; from and to *b*) are not completely equivalent, the (dis)similarity depends on the difference of the values (of ties) and on the (dis)similarity (in terms of regular equivalence) of units *a* and *b* compared relative to their magnitude. REGE solves the problem that regular equivalence does not demand that two units which are equivalent be connected to the same number of equivalent units, by allowing that different links of *a* can be matched by the same link of *b*. It should be mentioned that all REGE algorithms are designed to find only the maximal regular partition and not other regular partitions (Žiberna, 2008).

As mentioned above, all REGE algorithms are iterative algorithms. In the simulations presented in this paper, the number of iterations was set to 3, while an effect using more iterations was also tested later.

[13] No measure exists that can be computed in closed form.

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, Everett and Freeman, 1999) uses single linkage hierarchical clustering to find an appropriate partition. However, based on my experience and results presented by Žiberna (2008), Ward's (1963) clustering algorithm is used in this paper.

In addition to methods designed for regular equivalence, methods for structural equivalence were also used on the same dataset. Within indirect approach the method that was used was Ward's hierarchical clustering on distances computed using the Corrected Euclidean-like dissimilarity (Burt and Minor, 1983 in Batagelj, Ferligoj, and Doreian, 1992) with $p = 2$.

The above described methods are indicated in the results as:

- **sedist|str** – Direct approach using structural equivalence
- **REGGE, REGGE-OW, REGDI, REGDI-OW** – The version of REGE used.

4 Design of simulations

All simulations were done by repeating the following general procedure for a given number of times for each setting:

1. Generation of a valued network based on a blockmodel and other parameters
2. Application of different methods
3. Comparing the obtained partition(s) with the original partition using the Adjusted Rand Index (Hubert and Arabie, 1985: 198)

The way networks are generated has a large effect on the performance of evaluated approaches, so this is of great importance. The selected way of generating networks was constructed based on a review of numerous social networks, in particular networks of primary school pupils collected by Zemljič and Hlebec (2001).

The networks were generated based on the following parameters:

- partition
- blockmodel
- parameter controlling the enforcement of strict regularity in the binary network
- 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 described in Appendix 1. The meaning of the above parameters also becomes clearer when their

values in the individual settings are presented in the following sections. The limitations of the results due to the way the networks were generated are discussed in Section 6.

The result of each blockmodeling method is a partition (or sometimes a set of partitions). These were compared with the original partition used in the generation of the networks using the Adjusted Rand Index (Hubert and Arabie, 1985: 198). The so computed Adjusted Rand Index 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 several[14] networks were simulated and the same number of 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 optimization process. In such cases, the mean of the Adjusted Rand Indices for the partitions obtained on each network was first computed[15]. Then, the mean of these Adjusted Rand Indices over all networks was computed for each method.

The simulations were accomplished in two stages. Stage 1 was used as a preliminary stage, where a larger number of different settings and other options were tested on very small networks with only 11 units. In Stage 2, the methods were evaluated on more reasonably sized networks of 25 units, while the number of different settings was reduced. Larger networks were not generated due to the time required to run the simulations.

The simulations were conducted based on 25 settings that differed in the way the networks were generated, that is in the values of the parameters *partition, blockmodel, parameters of the beta distribution* and *multiplication factor*. Each setting could be performed with regularity enforced or not. However, a number of characteristics of the network generating procedure were also the same for all (or most) settings. These characteristics are described first.

The basic characteristics are:

- The number of units is 11 in Stage 1 and 25 in Stage 2.
- The procedure (generation of a network, application of methods, etc.) was repeated 20 times for each setting. In Stage 2 more simulations were conducted (when possible due to the availability of commuters) to obtain more accurate estimates (e.g. of the mean Adjusted Rand Index). In some cases, up to 100 repetitions were used.
- Two different partitions were used in each stage:
  - Stage1:

[14] For each settings at lest 20 networks were generated. In Stage 2, up to 100 networks were generated per setting.

[15] In the case where 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.

    - Two-cluster partition: 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2 and
    - Three-cluster partition: 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3
  - Stage1:
    - 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 and
    - Three-cluster partition: 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3
- *shape1* parameter has values 10, 8, 6, 4, and 2 in Stage 1 and values 8 and 4 in Stage 2. However, even in for Stage 1 only results for *shape1* values 8 and 4 are reported.
- Optimizational procedure:
  - Stage 1: 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.
  - Stage 2: 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 initially set to 3. The effect of using 100 iterations was also tested in additional iterations.
- In some settings in Stage 1, the block size normalization was tested on all generalized blockmodeling methods and maximum normalization on implicit blockmodeling. The selected settings are indicated by '**Norm**' in Table 1 in the "*Stage 1 – add. methods"* column**.**
- The methods (and additional settings) used can be divided into three groups in terms of their use in Stage 2. The markings in bold at the start of the paragraphs are used in Table 1 (least two columns) to indicate which methods (and additional settings) were used on a given setting in Stage 2. If a setting does not have any markings in the last two columns, this setting was not used in Stage 2. Used methods are indicated separately for two variations of each setting – when regularity was enforced when generating network and when it was not enforced.
  - **C** - Common methods – methods used in all settings in Stage 2. These methods include both direct and indirect approaches. The indirect approaches are the 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 3.1): 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|mean, ss|reg|max, REGDI, REGDI-OW, REGGE, and REGGE-OW.

- **P** - 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** - 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 (or those not ambiguously defined in the above list) used in generation of the networks are shown in Table 1. The settings presented in Table 1 are referred to by their "names", which are composed by the values of parameters *k, blockmodel, shape2* and *mf* pasted together by the "|" character.

Values used to describe the blockmodel and parameters *shape2* and *mf* in Table 1(when not simple numbers) are explained in Table 2. When the value in Table 1 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 max. regular (Not maximal regular), Diff. dist. par. (Different distribution parameters),* and *Diff. 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 1: Settings used in the simulations.

Id	***k***	**blockmodel***	***shape2* ***	***mf* ***	***settings class***	***Stage 1 – add. methods***	**Stage2**	
							regularity enforced	**regularity not enforced**
1	2	CP	1	10	Not max. reg.	Norm	C, P, 100	C, P, 100
2	2	CP	4	10	Not max. reg.			
3	2	1T	1	10	Clear pattern	Norm	C, P, 100	C, P, 100
4	2	1T	4	10	Clear pattern		C, P	C
5	3	C2P	G3	10	Diff. dist. par.			
6	3	C2P	4	G3	Not max. reg.			
7	3	2T	G3	10	Diff. dist. par.			
8	2	BG	4	10	Not max. reg.		C	C
9	2	BG	1	10	Not max. reg.	Norm	C, P, 100	C, P, 100
10	2	CP	D	10	Diff. dist. par.	Norm	C, P	C, P
11	2	CP	R	10	Diff. dist. par.		C	
12	2	BG	R	10	Diff. dist. par.	Norm	C, P	C, P
13	2	CP	1	D	Diff. block max.		C	C
14	2	CP	1	R	Diff. block max.	Norm		
15	2	BG	1	R	Diff. block max.		C, P	C, P
16	2	AR	1	D	Diff. block max.	Norm	C	C
17	2	AR	1	R	Diff. block max.		C	C
18	2	AR	D	10	Diff. dist. par.	Norm	C	
19	2	AR	R	10	Diff. dist. par.		C	
20	3	G3	G2	10	Diff. dist. par.			
22	3	2V1	G1	10	Diff. dist. par.		C	C
23	3	13	1	10	Clear pattern		C, P, 100	C, P
24	3	C	O4	10	Diff. dist. par.		C	C
25	3	C	1	10	Not max. reg.		C, P, 100	C, P

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 shown in Table 2.

Table 2: Values (codes) in Table 1.

blockmodel

BG (Between Groups):
```
     [,1]   [,2]
[1,] "null" "reg"
[2,] "reg"
"null"
```

CP (Core-Periphery):
```
     [,1]  [,2]
[1,] "reg" "reg"
[2,] "reg" "null"
```

1T (1 Tie):
```
     [,1]   [,2]
[1,] "null" "reg"
[2,] "null"
"null"
```

AR (All Regular):
```
     [,1]  [,2]
[1,] "reg" "reg"
[2,] "reg" "reg"
```

C2P (Core and 2 Peripheries):
```
     [,1]  [,2]   [,3]
[1,] "reg" "reg"  "reg"
[2,] "reg" "null" "null"
[3,] "reg" "null" "null"
```

2T (2 Ties):
```
     [,1]   [,2]   [,3]
[1,] "null" "reg"  "reg"
[2,] "null" "null" "null"
[3,] "null" "null" "null"
```

2V1 (2 Versus 1):
```
     [,1]   [,2]   [,3]
[1,] "reg"  "reg"  "null"
[2,] "reg"  "reg"  "null"
[3,] "null" "null" "reg"
```

G3 (Group 3 different - disconnected):
```
     [,1]   [,2]   [,3]
[1,] "reg"  "reg"  "null"
[2,] "reg"  "reg"  "null"
[3,] "null" "null" "null"
```

13 (Tie from 1 to 3):
```
     [,1]   [,2]   [,3]
[1,] "null" "null" "reg"
[2,] "null" "null" "null"
[3,] "null" "null" "null"
```

C (Cycle):
```
     [,1]   [,2]   [,3]
[1,] "null" "reg"  "null"
[2,] "null" "null" "reg"
[3,] "reg"  "null" "null"
```

shape2 when block-specific

R (Row):
```
     [,1] [,2]
[1,]    4    4
[2,]    1    1
```

D (Diagonal):
```
     [,1] [,2]
[1,]    1    4
[2,]    4    1
```

O4 (One tie with value 4):
```
     [,1][,2][,3]
[1,]    1   1   1
[2,]    1   1   4
[3,]    1   1   1
```

G3 (Group 3 different) :
```
     [,1][,2][,3]
[1,]    1   1   4
[2,]    1   1   4
[3,]    4   4   4
```

G2 (Group 2 different) :
```
     [,1][,2][,3]
[1,]    1   4   1
[2,]    4   4   4
[3,]    1   4   1
```

G1 (Group 1 different) :
```
     [,1][,2][,3]
[1,]    1   4   4
[2,]    4   4   4
[3,]    4   4   4
```

mf (multiplication factor) when block-specific

D (Diagonal):
```
     [,1] [,2]
[1,]   10    6
[2,]    6   10
```

R (Row):
```
     [,1] [,2]
[1,]    6    6
[2,]   10   10
```

G3 (Group 3 different) :
```
     [,1][,2][,3]
[1,]   10  10   6
[2,]   10  10   6
[3,]    6   6   6
```

5 Results

In Stage 1, blockmodeling methods were evaluated on networks with only 11 units. This can be thought of as a preliminary stage where several ideas were tested. Therefore, it was possible to consider a wider range of settings and methods. In addition, due to a very limited number of different partitions of 11 units into two clusters, a full search was possible.

As Stage 1 was really meant only as a preliminary stage, for most settings only the main findings are reported here (see Žiberna (2007b) for all results). More detailed results are only reported for settings that differ significantly from those used in Stage 2. These are special settings that were designed to test the effects of normalizations of inconsistencies on the generalized blockmodeling approaches.

Stage 2 was designed to be more realistic since networks of 25 units were generated and analyzed, however on a smaller number of settings. As it was evident from the results of Stage 1 that the relative performance of the methods is similar for different values of the *shape*1 parameter, the simulations were run only using two values of this parameter, namely, 8 and 4.

5.1 Figures used for representing results

In this paper, figures are used to present the results of 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, which define the way that the networks in the simulations are generated. The label on the axis is comprised of the information in columns 2 to 5 of Table 1, 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 *shape1* parameter. On the y axis, some statistic is usually represented. In most figures (including Figure 1), this statistic is the mean of the Adjusted Rand Indices computed as described in Section 4.

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 within 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 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 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.

5.2 Results for common methods

In this subsection only results for Stage 2 are presented in detail. The results of Stage 1 will only be presented in more detail in Subsection 0 where some modifications of the methods were tested. Other results of Stage 1 are only included as comments next to the results of Stage 2 if they differ significantly. The results from Stage 2 for the common methods (as defined in Section 4) are presented in Figure 1 for *shape1* = 8 and in Figure 2 for *shape1* = 4. In addition, the results of additional simulations testing the effect of using more iterations in the REGE algorithm are presented in Figure 3 for *shape*1 = 8 (the results for *shape*1 = 4 follow a similar pattern). It should be noted that, as the effect the number of iterations on REGE was tested in additional simulations, the results of these simulations (of REGE with more iterations) cannot be directly compared to the results of other approaches.

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 comparing the two methods used for structural equivalence, it is clear that the direct approach in most cases performs much better than the indirect approach and never worse.

Binary and valued blockmodeling (without pre-specified blockmodeling)**:** Binary blockmodeling performed very poorly (usually the worst of all methods) in all 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 as poorly as in other settings. Some ideas for the bad performance of these methods are presented at the end of this subsection. This is in large contrast to the results of Stage 1 where binary blockmodeling performed well especially with the *Clear pattern* settings with large values of *shape1* parameter. It also performed well in some *Not maximal regular* settings, where it even outperformed valued blockmodeling.

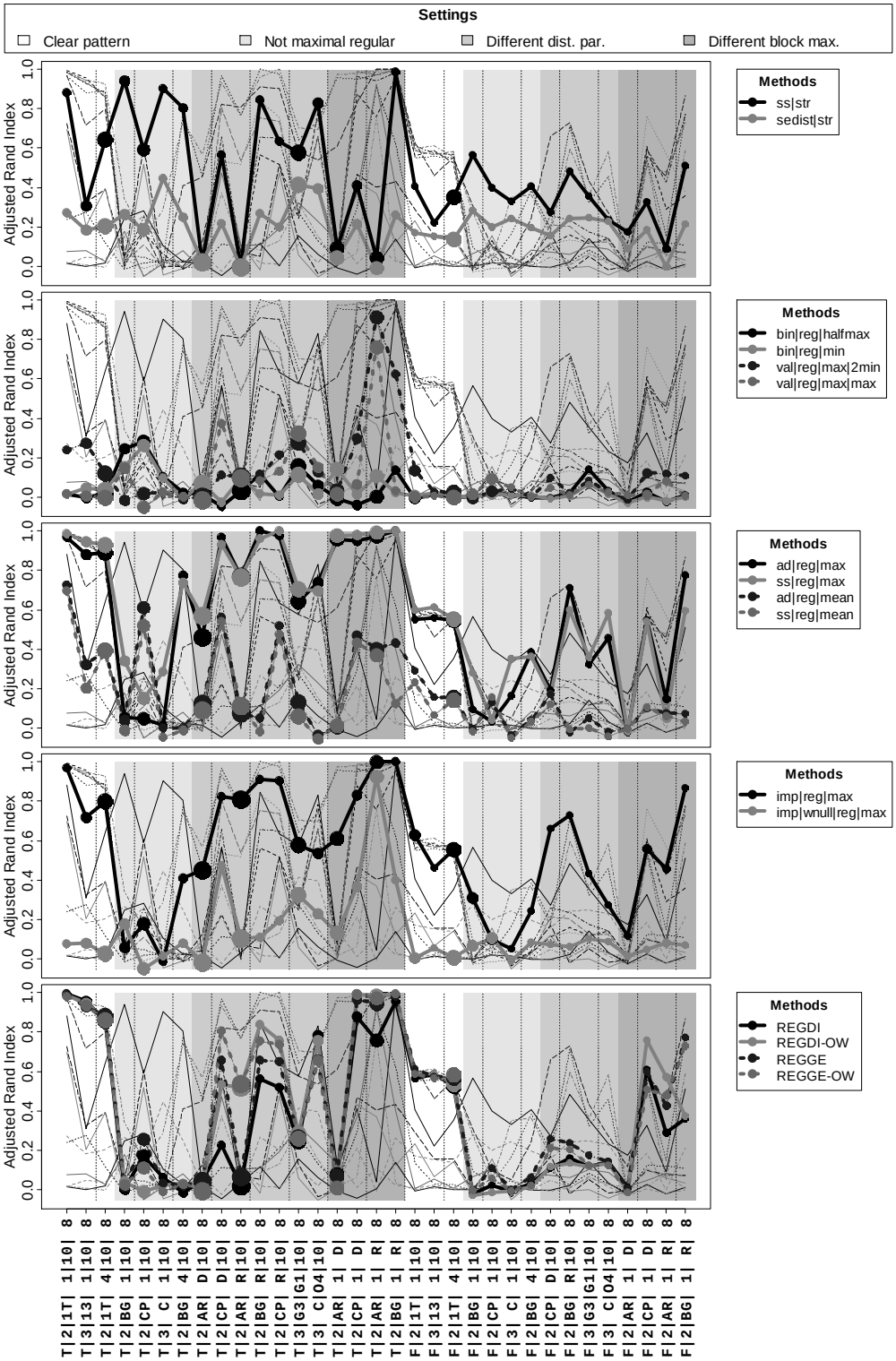


Figure 1: Results of simulations in Stage 2 for all settings with *shape1* = 8.

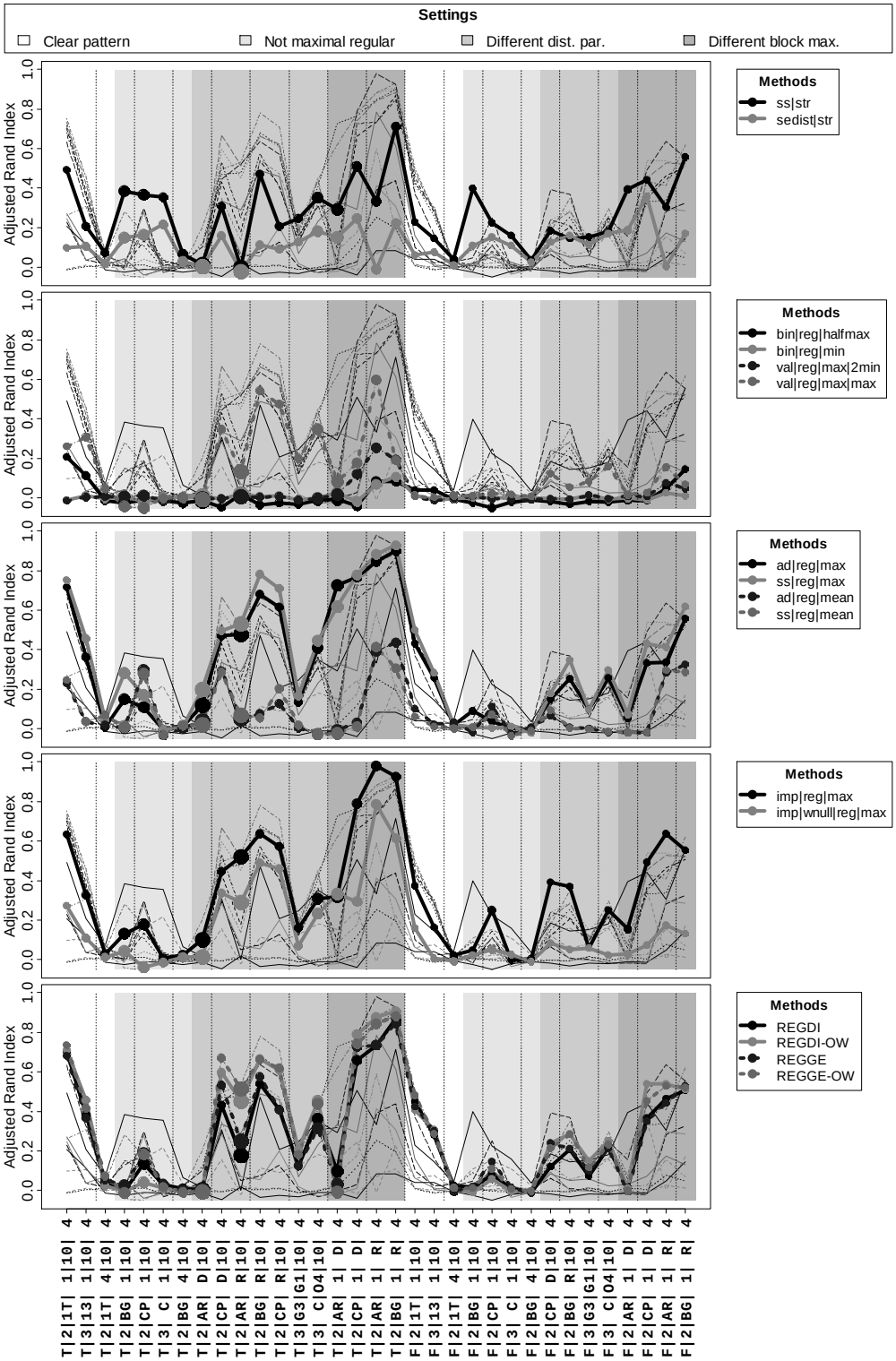
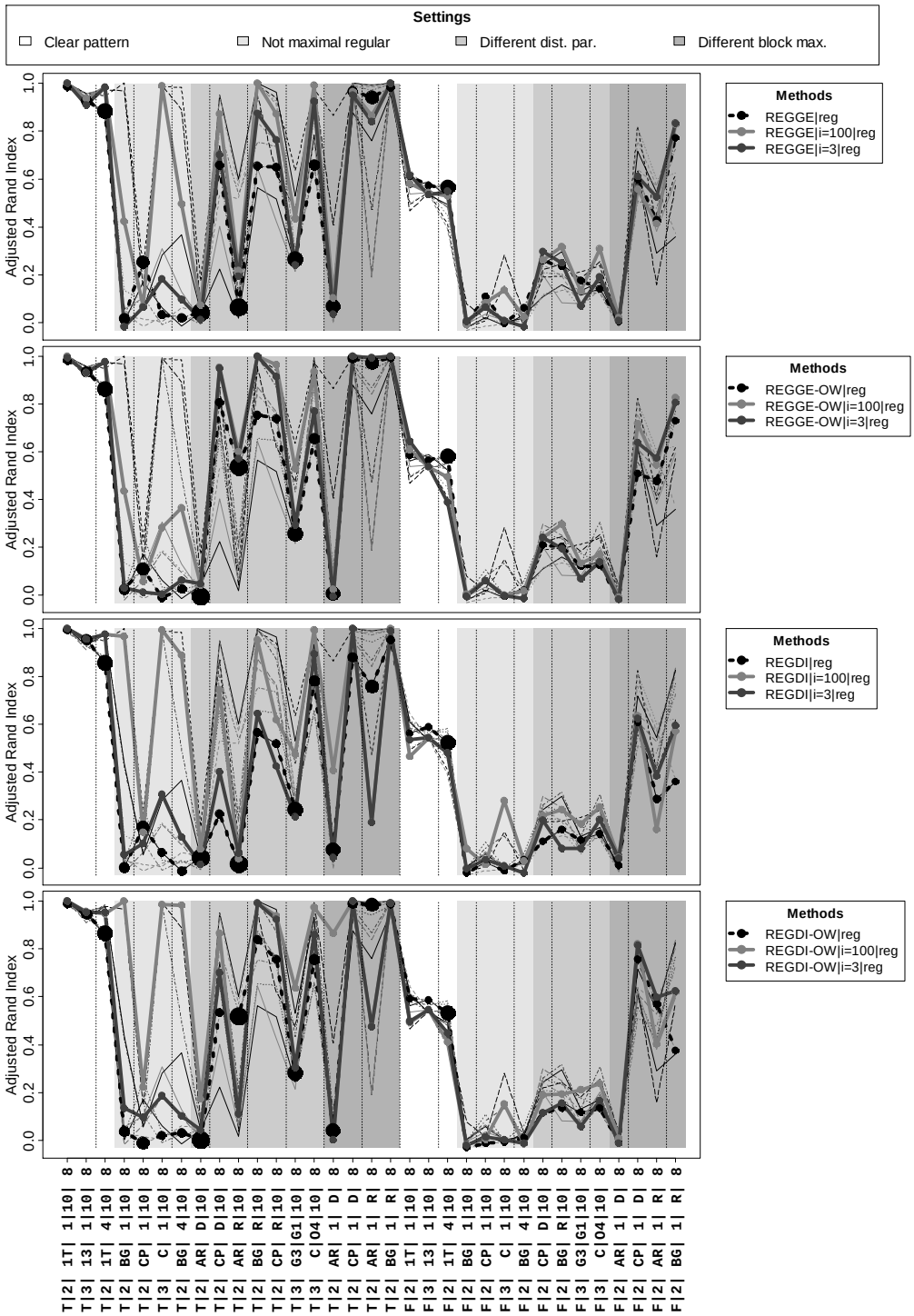


Figure 2: Results of simulations in Stage 2 for all settings with *shape1* = 4.



Legend: "i=x", where x is a number (3 or 100) indicating the number of iterations used in the REGE algorithm. If this is not present, the result is from the original Stage 2 simulations where the number of iterations was set to 3.

Figure 3: Testing the effect of the number of iterations in REGE.

Implicit blockmodeling (without pre-specified blockmodeling)**:** Implicit blockmodeling with regular blocks only performed similarly to homogeneity blockmodeling according to *max*-regular equivalence, although slightly worse in most settings. However, it also performed slightly better in some. 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. On the other hand, in Stage 1 the use of null blocks slightly improved the performance in some settings[16] and reduced it in some settings[17].

REGE with 3 iterations: REGE with 3 iterations 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 the settings where it performed well it usually performed slightly worse than homogeneity and implicit blockmodeling, especially when the *shape1* parameter was 8 and regularity was not enforced. All four versions performed similarly in most cases. Still, 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[18] networks (based on these partitions) for that REGE algorithm than to any other (REGE algorithm).

REGE with 100 iterations: As only 3 iterations might be inadequate, an additional test about the effect of the number of iterations was performed and the results for *shape1* = 8 are presented in Figure 3 (the results for *shape1* = 4 follow a similar pattern). Here, both results of the original simulations for REGE are presented together with the results of additional simulations for REGE with 3 and 100 iterations. The new results are not directly comparable with those for other approaches (as they are based on different simulation runs). However, we can assume that the effect of going from 3 to 100 iterations would have been similar even on those runs. With more iterations REGE becomes one of the best approaches (in particular REGDI-OW). What is very surprising is that even in some *Not maximal regular* settings REGE can perform very well if the number of iterations is increased (to 100), even so it seems that one of the REGE approaches (i.e. REGDI-OW) is the best approach in these settings[19]. Not only is this very strong effect surprising, but even more surprising is the fact that REGE algorithms perform well in such settings. This actually means

[16] E.g. settings 'T|3| G3|G2|10| 8', 'T|3| 2T|G3|10| 8' and 'T|3|C2P|G3|10| 8'.

[17] E.g. settings 'T|2| CP| 1|10| 8', 'T|2| CP| 4|10| 8' and 'T|2| AR| D|10| 8'.

[18] What kind of network is 'ideal' for most a versions of REGE algorithm is demonstrated in Žiberna (2008).

[19] As the effect the number of iterations on REGE was tested in additional simulations, the results of these simulations (of REGE with more iterations) cannot be directly compared to the results of other approaches.

that they do not perform as designed as they should find a maximal regular partition and that is, in these settings, a partition with all units in one cluster. This is actually completely true only for REGGE-OW since this is the only version of REGE that is totally compatible with the way networks were simulated in this study. Interestingly, for this version of REGE the effect of the number of iterations is the smallest. The good performance of REGGE and REGDI in the setting with blockmodel C is expected. The reason for the good performance of REGE lies in the fact that the networks generated are not ideal (they do not perfectly conform to regular equivalence, at least not when taking values into account). As these imperfections are relatively small, they need more iterations to have an effect. Further research is needed for a more precise explanation. We can, however, conclude that for such relatively small networks 3 iterations is enough for a network where REGE is expected to perform well; however, more iterations can improve the results for networks where the structure is unclear to REGE. When using 100 iterations REGE performs

Three 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 used without pre-specified blockmodels. The third one is the good performance of REGE with 100 iterations in settings *Not maximal regular*, as it was not designed to perform well in such settings.

One possible reason for the good performance of sum of squares blockmodeling according to structural equivalence 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 as possible 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 0 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) were found. The applications of binary blockmodeling to real valued networks (e.g. in Žiberna, 2007a, 2007b, 2008) have also shown its tendency to find several partitions as 'optimal' when used to find relatively 'loose' equivalences such as regular equivalence. The other factor is that the optimization procedure. As can be seen in Subsection 0, this factor 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). Figure 9 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).

5.3 Results for pre-specified blockmodeling

The results for pre-specified blockmodeling are presented in Figure 4 for *shape1* = 8 and in Figure 5 for *shape1* = 4. As indicated in Table 1, pre-specified blockmodeling was used in only some settings. 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 and where implicit blockmodeling (among the settings where regularity was enforced) 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. In Stage 1 these approaches (with pre-specified blockmodeling) were even among the best approaches in almost all settings.

5.4 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 approaches and maximum normalization on implicit blockmodeling. These simulations were made as part of Stage 1 (smaller networks) using only a selection of settings as indicated in Table 1 with the marking **Norm**. An attempt was made to select settings as diverse as possible, while not using those settings where the *shape2* parameter was set to four in all blocks.

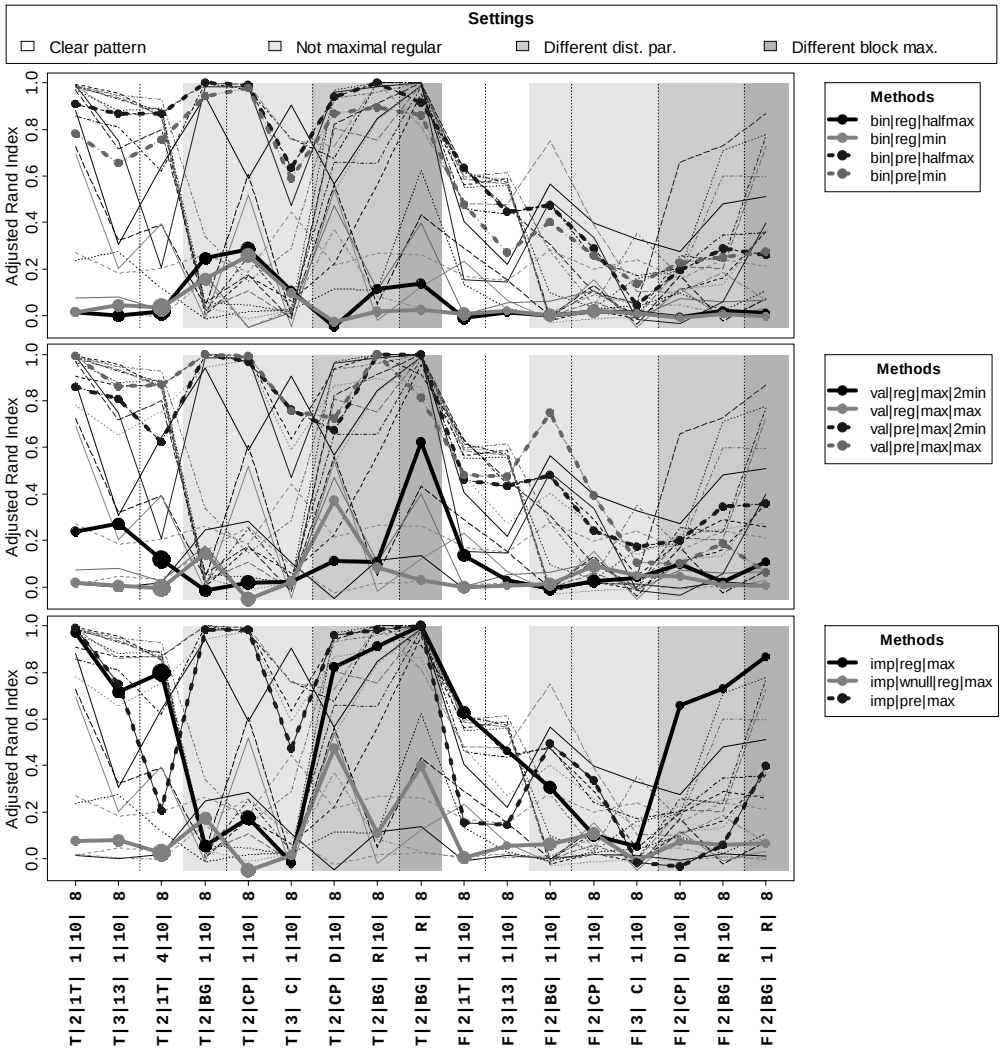


Figure 4: Results for pre-specified blockmodeling for selected settings where parameter *shape1* = 8.

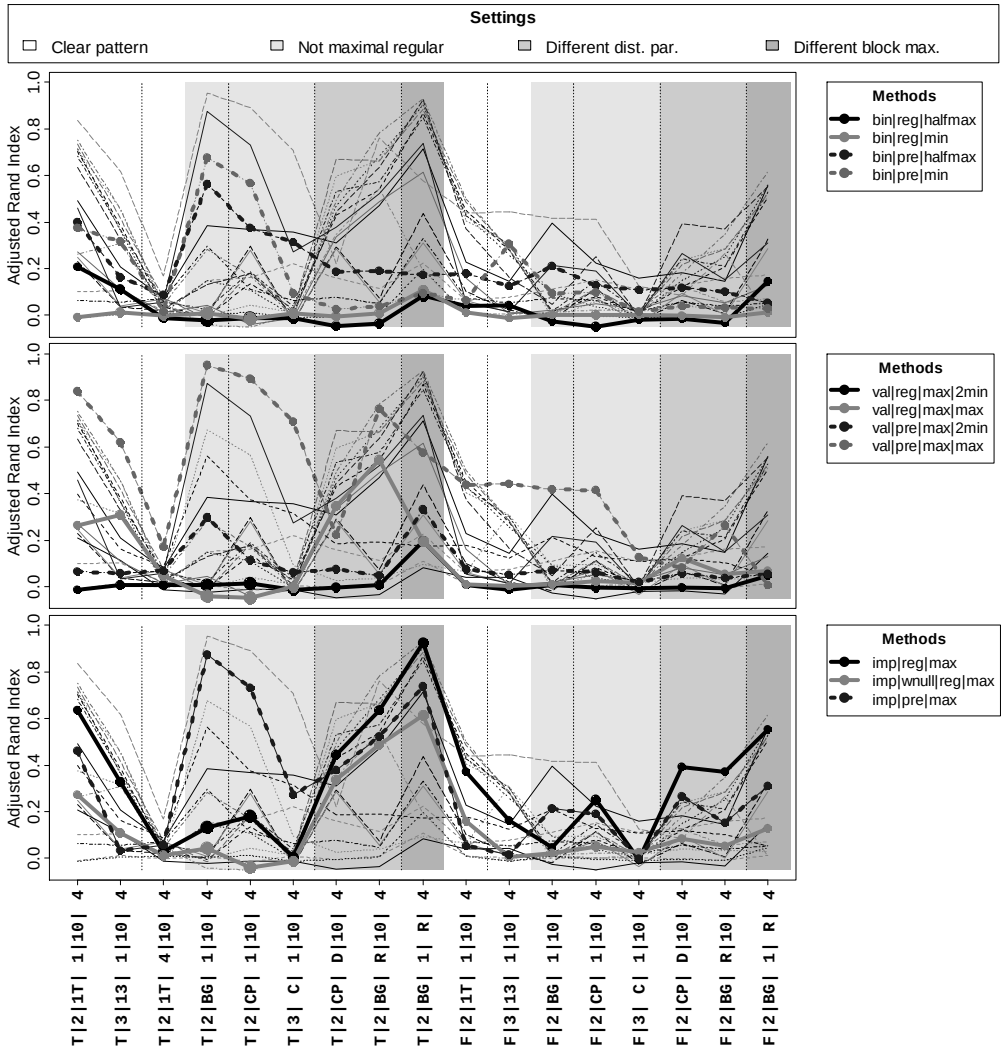


Figure 5: Results for pre-specified blockmodeling for selected settings where parameter *shape1* = 4.

The results that show the effect of block size normalization are shown in Figure 6 for *shape1* = 8. The effect was also tested for *shape1* = 4; however, the results are not shown due to space limitations. 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|1T| 1|10`** and **`2|CP| D|10`**, while it is usually positive in setting **`F|2|AR| D|10`**.

Therefore, the use of maximum normalization is not advised. Regarding the block size normalization, no clear advice can be given. 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.

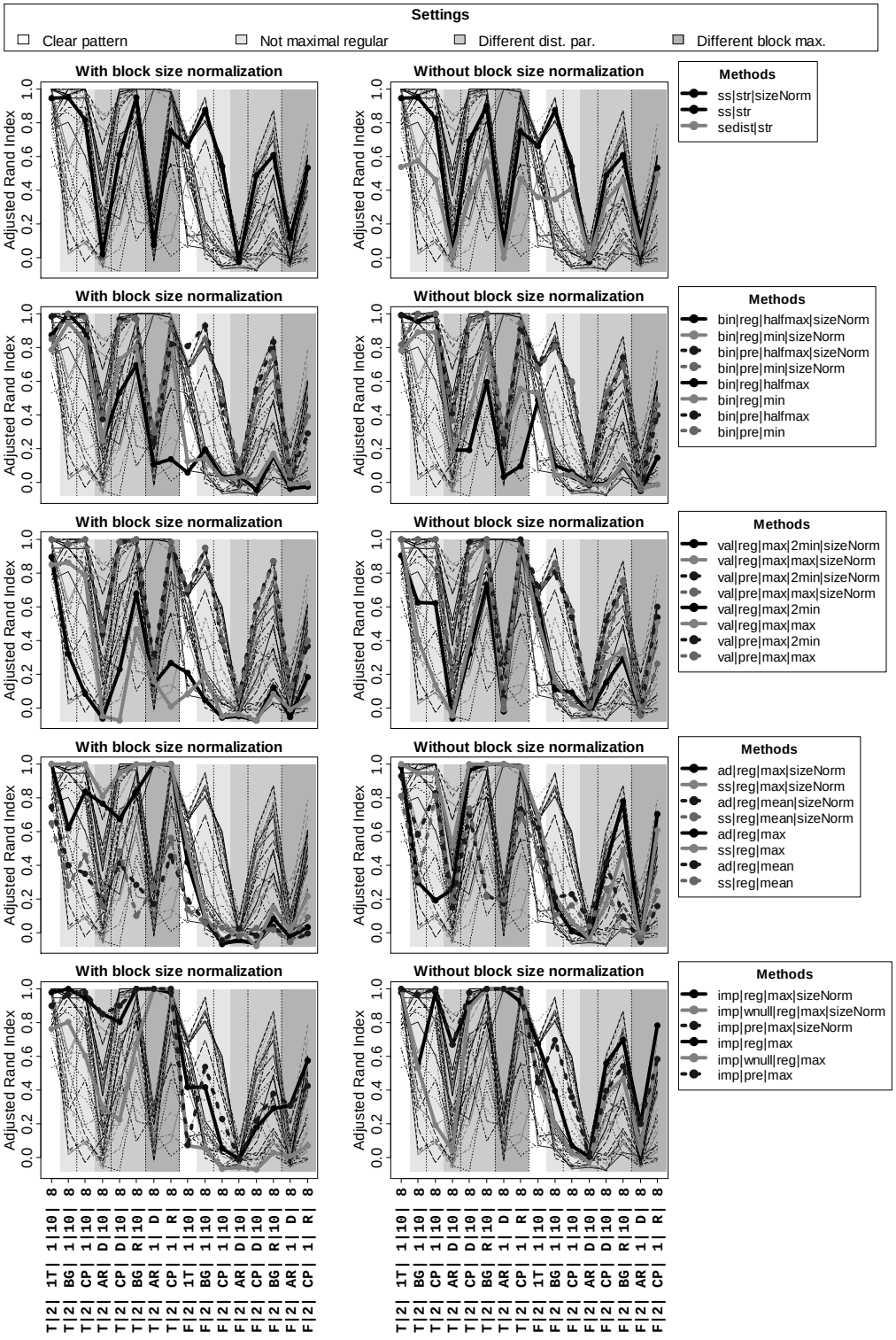


Figure 6: Effects of block size normalization for selected settings with *shape1* = 8.

5.5 Evaluation of optimization

The results obtained using the direct approach presented up till now 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 of direct approaches (when it occurred), especially the very poor performance of binary and valued blockmodeling, might be caused by the fact that the optimization of 20 random starting partitions finds only locally but not globally optimal partitions.

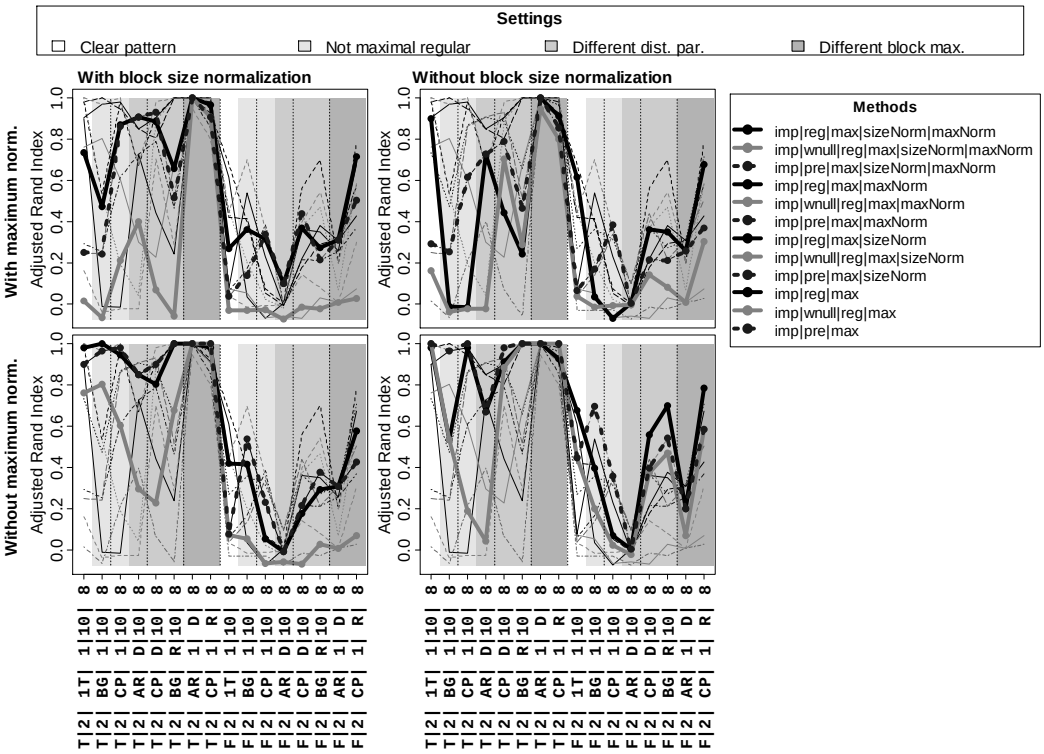


Figure 7: Effects of block size and maximum normalization (and their interaction) on implicit blockmodeling for selected settings with *shape1* = 8.

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 1) 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 obtained using optimization of 20 random starting partitions are compared with those obtained using the optimization of 100 starting partitions and with those obtained with an optimization of the *correct* partition (the one used in generation of the networks).

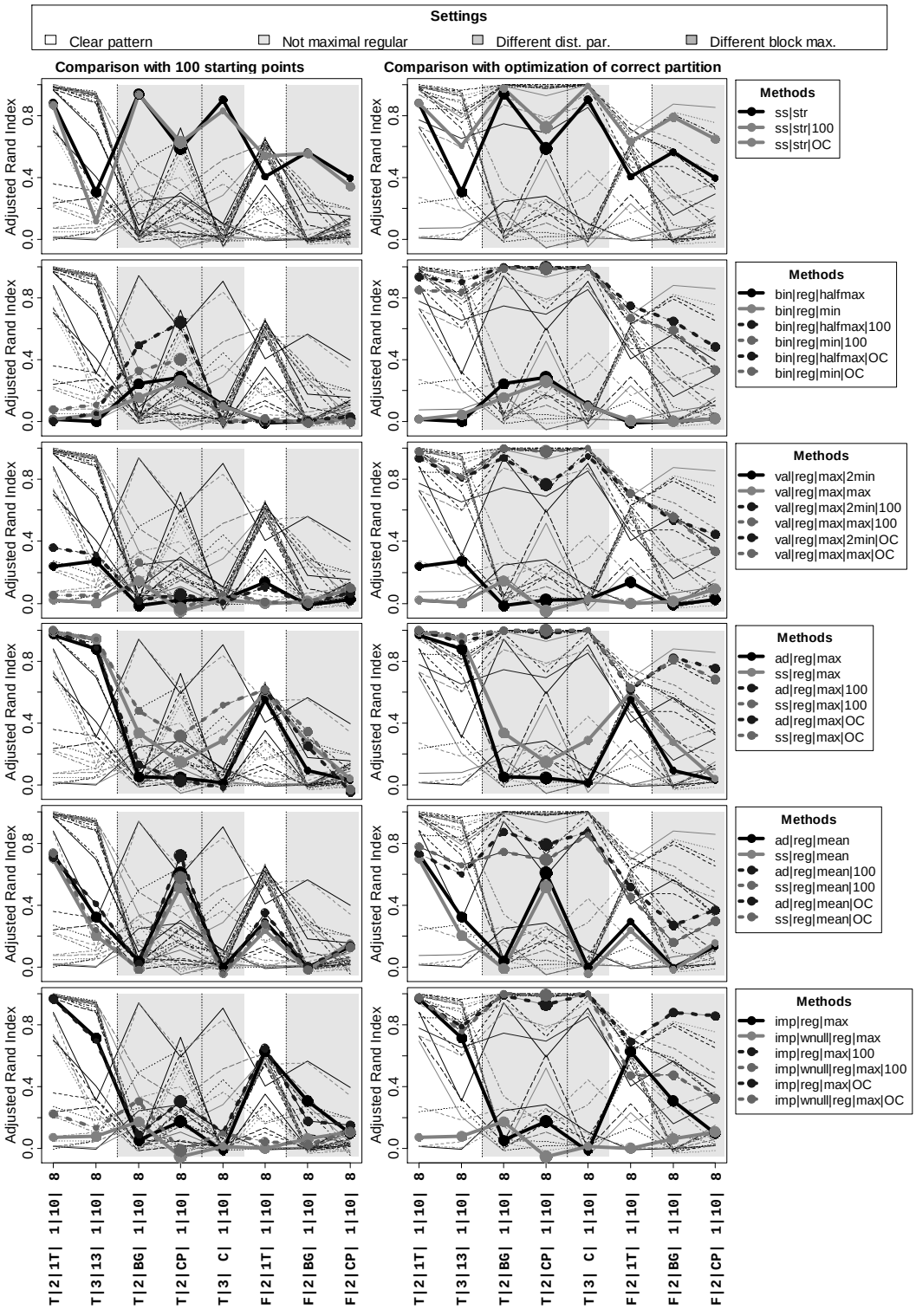


Figure 8: Results obtained by a local optimization of 20 and 100 random starting partitions and of the correct partition.

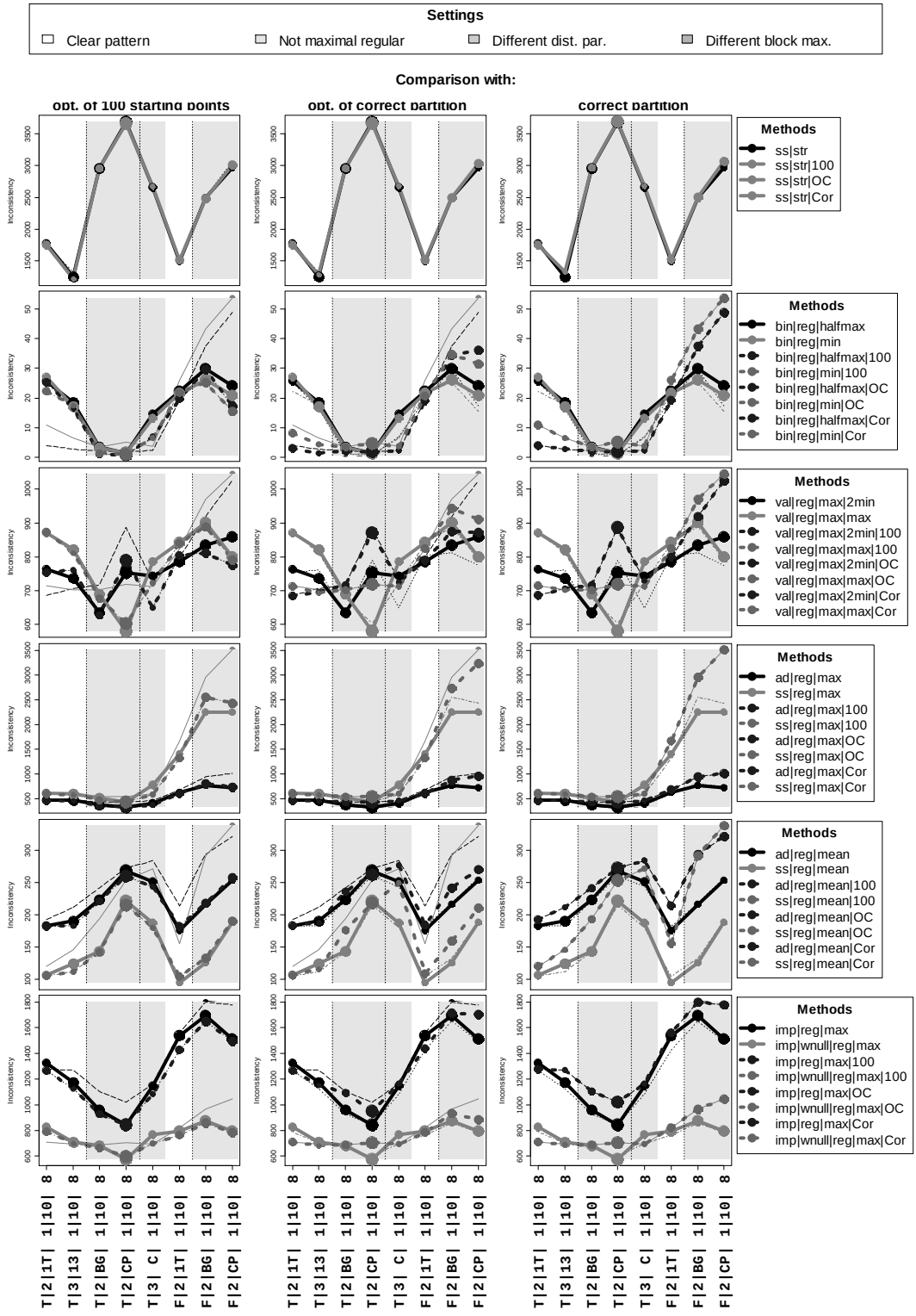


Figure 9: Comparison of inconsistencies.

The results are presented in Figure 8. However, here the comparison is not made only according to the Adjusted Rand Indices but also to the (total) inconsistencies[20]. 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 9.

The conclusions based on both measures 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 a possibility that and error has been made when producing these figures was even considered. However, a careful examination of the generated networks confirmed that all these relatively different partitions have very similar inconsistencies. This indicates that optimization was successful.

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 local (and not a global) 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.

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*

[20] 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.

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 also 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.

6 Limitations

The simulation study presented in this paper 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:

The size of the networks: Only networks of size 11 and 25 units were generated. Larger networks were not generated due to the time needed to analyze them using generalized blockmodeling of valued networks. This is a serious

limitation of the study for two reasons. First, the study aims at evaluating methods for detecting regular equivalence and therefore regular blocks. However, regular blocks are defined as such blocks where there is at least one tie (in the binary sense) in each row and in each column. When networks grow larger (and at fixed number of clusters, also blocks), this requirement can be fulfilled by increasingly sparser blocks. Therefore, they are presumably also much harder to detect. The second reason is an empirical one. When comparing results obtained on networks with 11 units with those with 25 units the performance of all methods considerably worsened, indicating that the size of the network definitively has an important impact on the results of the evaluation. I assume that increasing the size of the network further would have similar yet diminishing effect, although this should of course be tested with further simulations.

Simulation of the '0' ties: The '0' ties were simulated from the beta distribution described in Appendix 1. 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 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 affect of this assumption was tested in Stage 1 (results are presented in Žiberna (2007b, 187-188)) 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 *shape1* 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 a lot of the real social networks have 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. This problem is especially relevant for generalized blockmodeling methods. However, as the implementation of all generalized blockmodeling approaches suffers from the same problems listed below, which means that the results obtained in this paper 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 paper 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 in this paper did not have clusters of extremely different sizes, this shortcoming might still have some effect 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.

Another limitation of this study is that initially only 3 iterations were used in the REGE algorithms. However as this might be inadequate, we later tested the effect of using 100 iterations instead of 3. Based on these tests, we can conclude that for such relatively small networks 3 iterations is enough for a network where REGE is expected to perform well; however, more iterations can improve the results for networks where the structure is unclear to REGE. These results were also incorporated in the discussion of the main results and conclusion.

7 Conclusions

These conclusions are mainly based on the simulation of networks with 25 units. While networks with only 11 units were also analyzed, 11 units are too few to produce relevant results regarding regular equivalence. The difference between the results based on networks with 11 and those based on networks with 25 units is striking. Most methods, but especially binary and valued blockmodeling, performed considerably better on a smaller network. 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. The second one is there are much less possible partitions into two or three clusters of 11 units than of 25. Due to this, a full search was used when analyzing networks with 11 units 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 5.5) 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 networks generated 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. This doubt is also supported by some examples in Žiberna (2007b), as in most of them 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 5.2. Therefore, we can conclude that the good performance of methods for structural equivalence should be taken with caution, but not dismissed.

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, Ferligoj and Doreian, 1992) 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 homogeneity blockmodeling, REGE and (although somewhat less) implicit blockmodeling are well suited for identifying regular portions in valued networks (when regularity was enforced). Until now, the REGE algorithm (the direct approach) was the only method designed to find partitions in terms of regular equivalence in valued networks. We have shown that homogeneity blockmodeling performs at least as well as REGE. In addition, homogeneity blockmodeling like other methods of generalized blockmodeling can, unlike REGE, also be used to partition the network based on

other kinds of equivalences (other than regular). It can be used with other allowed block types (other than (*max*-)regular) and to some extent also with pre-specified blockmodels. REGE (with 100 iterations) performed well in all settings for which it was designed and in some where it was not supposed to, which itself calls for further investigation. It was expected that REGE would perform well only in settings where the partition that was searched for was maximal regular (at least when taking the values of ties into account) as this is what it was designed for. Implicit blockmodeling performed similarly as homogeneity blockmodeling and REGE, although slightly worse.

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 units). Especially in the case of binary blockmodeling, there are indications that the problem is in the search procedure as the correct partition usually had smaller 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.

8 Ideas for future work

The first set of ideas for further work of course deals with the limitations of the simulations presented in this paper, as discussed in Section 6. However, in addition to these more technical improvements, many questions that can also be answered using simulations are still open and are presented below.

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) 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.

References

[1] Batagelj, V., Doreian, P., and Ferligoj A. (1992): An optimizational approach to regular equivalence. *Social Networks*, **14**, 121–135.

[2] Batagelj, V. and Ferligoj, A. (2000): Clustering relational data. In W. Gaul, O. Opitz, M. Schader (Eds.): *Data Analysis*, 3 – 15. New York: Springer-Verlag.

[3] Batagelj, V., Ferligoj, A., and Doreian, P. (1992): Direct and indirect methods for structural equivalence. *Social Networks*, **14**, 63–90.

[4] Batagelj, V. and Mrvar A. (2006): *Pajek 1.11*, Available at http://vlado.fmf.uni-lj.si/pub/networks/pajek/ (January 6, 2006).

[5] Borgatti, S.P. and Everett, M.G. (1992): Regular blockmodels of multiway, multimode matrices. *Social Networks*, **14**, 91–120.

[6] Borgatti, S.P. and Everett, M.G. (1993): Two algorithms for computing regular equivalence. *Social Networks*, **15**, 361-376.

[7] Borgatti, S.P., Everett M.G., and Freeman L.C. (1999): *Ucinet 5 for Windows: Software for Social Network Analysis*. Natic: Analytic Technologies.

[8] Burt, R.S. and Minor M.J. (1983): *Applied Network Analysis*. Beverly Hills: Sage.

[9] Doreian, P. (2006): Some Open Problem Sets for Generalized Blockmodeling. In V. Batagelj, H.-H. Bock, A. Ferligoj, and A. Žiberna (Eds.): *Data Science and Classification*, 119-130. Berlin: Springer-Verlag.

[10] Doreian, P., Batagelj, V., and Ferligoj, A. (1994): Partitioning networks on generalized concepts of equivalence. *Journal of Mathematical Sociology*, **19**, 1–27.

[11] Doreian, P., Batagelj V., and Ferligoj A. (2005): *Generalized Blockmodeling*. New York: Cambridge University Press.

[12] Ferligoj, A. (1989): Razvrščanje v skupine, Teorija in uporaba v družboslovju. *Metodološki zvezki št. 4*, Ljubljana.

[13] Hubert, L. and Arabie P. (1985): Comparing partitions. *Journal of Classification*, **2**, 193-218.

[14] Luczkovich, J.J., Borgatti, S.P., Johnson J.C., and Everett M.G. (2003): Defining and measuring trophic role similarity in food webs using regular equivalence. *Journal of Theoretical Biology*, **220**, 303–321.

[15] Nordlund, C. (2007): Identifying regular blocks in valued networks: A heuristic applied to the St. Marks carbon flow data, and international trade in cereal products. *Social Networks*, **29**, 59-69.

[16] R Development Core Team (2006). *R: A Language and Environment for Statistical Computing*. Vienna: R Foundation for Statistical Computing. Available at http://www.R-project.org (December 12, 2006).

[17] Ward, J.H. (1963): Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, **58**, 236-244.

[18] White, D.R. (1985a): *Doug White's Regular Equivalence Program*. Available at http://eclectic.ss.uci.edu/~drwhite/REGGE/REGGE.FOR (May 12, 2005).

[19] White, D.R. (1985b): *Doug White's Regular Equivalence Program*. Available at http://eclectic.ss.uci.edu/~drwhite/REGGE/REGDI.FOR (May 12, 2005).

[20] White, D.R. and Reitz K.P. (1983): Graph and semigroup homomorphisms on networks of relations. *Social Networks*, **5**, 193-234.

[21] Zemljič, B. and Hlebec, V. (2001): Zanesljivost mer središčnosti in pomembnosti v socialnih omrežjih. *Družboslovne razprave*, **XVII, 37-38**, 191-212.

[22] Žiberna, A. (2006): `blockmodeling 0.1.2`: *An R package for Generalized and classical blockmodeling of valued networks*. Available at http://www2.arnes.si/~aziber4/blockmodeling/ (April 16, 2006).

[23] Žiberna, A. (2007a): Generalized blockmodeling of valued networks. *Social Networks*, **29**, 105-126.

[24] Žiberna, A. (2007b): *Generalized Blockmodeling of Valued Networks*. *Doctoral thesis*. Ljubljana. Available at http://www2.arnes.si/~aziber4/ blockmodeling/Dissertation-final.pdf (June 22, 2009).

[25] Žiberna, A. (2008): Direct and indirect approaches to blockmodeling of valued networks in terms of regular equivalence. *Journal of Mathematical Sociology*, **32**, 57-84.

Appendix 1: Procedure for generating a network

The networks were generated using the following procedure:

1. A binary network was generated based on a partition, a blockmodel, and the parameter controlling the enforcement of strict regularity.
   - the partition was used to split a (empty) network (a matrix of 0s) into blocks and to determine the size of the network
   - with blocks where the blockmodel indicated null blocks, nothing was changed
   - in regular blocks, each cell had a probability of becoming 1 equal to p:
     - $p = \frac{1}{\min(n_r, n_c) - 1}$, where:
     - nr is the number of rows in the blocks
     - nc is the number of columns in the blocks
   - 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. A valued network was generated based on the binary network and the remaining parameters (beta parameters and multiplication factor).
   - 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 Γ 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, *shape*1 and *shape*2.
     - For '0' ties, the values were generated from the beta distribution with parameter a always set to 1 and the parameter b set to *shape*1. For '1' ties, the values were generated from the beta distribution with parameter a set to *shape*1 and the parameter b set to *shape*2, which could be block specific. The parameter *shape*2 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 *shape*2 was 4. The parameter *shape*1 could take on values 10, 8, 6, 4 and 2, but was often restricted to valued 8 and 4. The values of the parameters *shape*1 and *shape*2 in individual settings are specified in Subsection 5.1 for Stage 1 and in Subsection 6.1 for Stage 2.
   - 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.