Sociological Methods & Research
40(1) 57–87

DOI: 10.1177/0049124110384947
http://smr.sagepub.com

Two Algorithms for Relaxed Structural Balance Partitioning: Linking Theory, Models, and Data to Understand Social Network Phenomena

Michael Brusco[1], Patrick Doreian[2,3], Andrej Mrvar[3], Douglas Steinley[4]

Abstract

Understanding social phenomena with the help of mathematical models requires a coherent combination of theory, models, and data together with using valid data analytic methods. The study of social networks through the use of mathematical models is no exception. The intuitions of structural balance were formalized and led to a pair of remarkable theorems giving the nature of partition structures for balanced signed networks. Algorithms for partitioning signed networks, informed by these formal results, were developed and applied empirically. More recently, "structural balance" was generalized to "relaxed structural balance," and a modified partitioning algorithm was proposed. Given the critical interplay of theory, models, and data, it is important that methods for the partitioning of signed

[1]Florida State University, Tallassee, FL, USA
[2]University of Pittsburgh, PA, USA
[3]University of Ljubljana, Slovenia
[4]University of Missouri–Columbia, MO, USA

Corresponding Author:
Patrick Doreian, Department of Sociology, University of Pittsburgh, 2G03 Forbes Quad, Pittsburgh, PA 25260, USA
Email: pitpat@pitt.edu

networks in terms of relaxed structural balance model are appropriate. The authors consider two algorithms for establishing partitions of signed networks in terms of relaxed structural balance. One is an older heuristic relocation algorithm, and the other is a new exact solution procedure. The former can be used both inductively and deductively. When used deductively, this requires some prespecification incorporating substantive insights. The new branch-and-bound algorithm is used inductively and requires no prespecification of an image matrix in terms of ideal blocks. Both procedures are demonstrated using several examples from the literature, and their contributions are discussed. Together, the two algorithms provide a sound foundation for partitioning signed networks and yield optimal partitions. Issues of network size and density are considered in terms of their consequences for algorithm performance.

Keywords

algorithms, blockmodeling, relaxed structural balance, signed networks

Leik and Meeker (1975) describe a ''theory-model-data triangle'' in their characterization of mathematical sociology. The items in this triangle are joined through both deductive and inductive links. Theoretical ideas, model formulation, and data analyses are intertwined to promote understanding of social phenomena. One important implication is that these three items need to be coherently related so that each informs and extends the others. Failure in any one area, be it theory, models, or data analytic methods, severely limits the credibility of empirical results. With social scientists paying greater attention to social network approaches, it is critical that theory, models, and data remain consistent when studying network phenomena. As an example, the important conceptualization of structural equivalence by Lorrain and White (1971) inspired major changes in how social networks were viewed and analyzed. Breiger, Boorman, and Arabie (1975) provided a practical blockmodeling method for clustering network data with an algorithm they called CONCOR for operationalizing and implementing structural equivalence. Burt (1976) proposed STRUCTURE as an alternative algorithm for delineating network structure using a different operationalization of structural equivalence. White and Reitz (1983), by building on the insights of Sailer (1978), formulated regular equivalence as a generalization of structural equivalence. Both types of equivalence and algorithms are part of the widely used package UCINET (Borgatti, Everett, and Freeman 2002). All of these ideas were

incorporated by Doreian, Batagelj, and Ferligoj (2005) into generalized blockmodeling as a general way of blockmodeling network structures in ways going well beyond using structural and regular equivalence. Their ideas have been fully implemented in Pajek[1] (Batagelj and Mrvar 1998). Yet, there is no consensus regarding blockmodeling, and different analysts can (and do) use these different algorithms for blockmodeling networks to establish different partitions of a specific network. In terms of Leik and Meeker's conceptual triangle, there has been a proliferation of methods in the data component that have not been tied fully into the theory and model components.

We look at one part of the blockmodeling literature devoted to analyzing *signed* networks and examine two seemingly different algorithms for partitioning signed social networks that turn out to provide consistent partitions and fully complement each other. Together, they ensure greater coherence for joining the theory, model, and method components in Leik and Meeker's triangle. Equally important, they provide greater confidence in established partitions of signed social networks.

Partitioning Signed Social Networks

Some social relations, for example like/dislike, esteem/disesteem, support/oppose, friend/enemy, have signs as an intrinsic feature. Much of the study, both empirical and conceptual, of these relations and the networks they form has been informed by the formalization of Heider's (1946) theory by Harary (1953) and Cartwright and Harary (1956). The formalization of Heider's insights concerning ''structural balance'' as a psychological process resulted in a remarkable theorem about the overall structure of a signed network that is consistent with structural balance. In formal terms, we define a network, (G, σ, μ), as an ordered triple where

(i) $G = (\mathcal{V}, \mathcal{A})$ is a valued digraph having $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$ as a set of m vertices[2] (units) and $\mathcal{A}$ as set of arcs with $\mathcal{A} \in \mathcal{V} \times \mathcal{V}$;
(ii) σ: $\mathcal{A} \rightarrow \{p, n\}$ is a sign function so that the arcs with sign p are positive and the arcs with sign n are negative[3]; and
(ii) μ: $\mathcal{A} \rightarrow \Re^+$ maps the arcs to the positive real numbers where the values represent the strength of the ties represented by the elements of $\mathcal{A}$. If a signed network is binary (where the arcs are either +1 or –1), this third definitional component is not needed.

Let a_{ij} denote an arc from v_i to v_j. A semiwalk of a signed directed network is an alternating sequence of vertices and arcs: $\{v_1, (a_{12}$ or $a_{21}), v_2, (a_{23}$ or $a_{32}), v_3, \ldots v_{n-1}, (a_{m-1m}$ or $a_{mm-1}), v_m\}$ where vertices and arcs can be

repeated. The sign of a semiwalk is the product of the signs of the arcs it contains. A semiwalk of a signed directed network is positive if and only if it contains an even number of negative arcs and is negative if and only if it has an odd number of negative arcs. A signed network (G, σ, μ) is defined as balanced if every closed semiwalk is positive. Cartwright and Harary (1956) proved that for a balanced signed network, (G, σ, μ), the set of vertices, $\mathcal{V}$, can be partitioned into two subsets (clusters) so that every positive arc joins units of the same subset and every negative arc joins units of different subsets.[4] By defining a signed network as ''clusterable'' when it contains no semiwalk with exactly one negative arc, Davis (1967) extended Cartwright and Harary's result by proving that for a clusterable signed network, (G, σ, μ), the set of vertices, $\mathcal{V}$, can be partitioned into *two or more* subsets (clusters) so that every positive arc joins units of the same subset and every negative arc joins units of different subsets. Davis coined the term *plus-sets* for sets of actors having only positive ties among themselves and only negative ties to outsiders.

The number of clusters in these partitions is denoted by K. For $K = 2$, we have the first of these ''structure theorems'' and $K \geq 2$ for the second. In general, we refer to the K balance of a signed network and consider a signed network to be K balanced if it has a partition structure consistent with either of these theorems. However, $K > 2$ is most relevant for empirical signed networks. When the vertices are partitioned into K clusters, called *positions* in the blockmodeling literature, the set of vertices is partitioned into K^2 blocks, which form a $K \times K$ *image matrix*. A block is positive if it contains positive arcs but no negative arcs and negative if it contains negative arcs but no positive arcs. Both block types can contain zeros.

When examining signed networks, there have been two primary concerns: (1) measuring the amount of imbalance in the network (as departures from the ideal structure described by the structure theorems) and (2) describing the partition structure (or structures) that is closest to the ideal forms implied by the structure theorems. Doreian and Mrvar (1996) suggested using a line index proposed by Harary, Norman, and Cartwright (1965) as a measure of imbalance and proposed a partitioning algorithm based on the structure theorems that minimize this measure. This heuristic algorithm is described in the next section. Their approach was incorporated into the generalized blockmodeling approach of Doreian et al. (2005). One idea in generalized blockmodeling is translating a type of equivalence into a set of permitted block types in a blockmodel. The partition structure of a K-balanced network implies an obvious blockmodel structure: positive blocks on the main diagonal of the image matrix and negative blocks off the main diagonal.[5]

Structural balance need not be the only operative process in shaping a network. Some actors may be regarded positively by many other actors who otherwise divide into mutually hostile subgroups. This differential popularity (Feld and Elmore 1982) cannot be accommodated within structural balance partitioning because it implies positive blocks off the main diagonal. Further, given a division of a network when there are clusters whose ties are mostly consistent with the balance hypotheses, other actors might adopt a mediating role for these mutually hostile subgroups. If present in a network, this also would imply positive blocks off the main diagonal. Finally, there can be subsets of actors who are mutually hostile, and if present, this would imply negative blocks on the diagonal of a blockmodel of a signed network. Doreian and Mrvar (2009) proposed a relaxed structural balance blockmodel where there are both positive and negative blocks, as is the case for balance, but allowed these blocks to appear anywhere in the blockmodel. They adapted their 1996 algorithm to accommodate this specification change. When applied to some extant signed networks, the refitted blockmodels provided a more nuanced interpretation of the structures delineated and generally had much better overall fits with the empirical data than for structural balance.

Doreian and Mrvar's (2009) generalization of structural balance theory led to a version of the theory that was labeled ''relaxed structural balance.'' It is a formal generalization of the former. Relative to structural balance, this is a change in the formal model underlying the theory–model–data triad. The formal generalization was motivated by a need to incorporate additional types of social psychological processes driving the change of signed network ties. Both structural balance and relaxed structural balance assume that the operation of social psychological processes leave distinctive traces in the observed sociometric structure that change over time. Any ''tendency toward balance'' under structural balance posits movement to a blockmodel structure with positive blocks on the main diagonal of the image matrix and negative blocks off the main diagonal. Correspondingly, any tendency toward balance under relaxed structural balance will result in a more complex set of potential image matrices. As such, it is a parallel change in the theory part of the theory–model–data triad. As a result, it is not surprising that this implies also a change in the data part of the theory–model–data triangle. The number of valid ideal-type blockmodel image structures was expanded, and it is important that methods used to identify sociometric structure are able to delineate, and discriminate between, these images. Any algorithms for fitting these new blockmodels have to be constructed to preserve the consistency of the theory–model–data triad. Given that the inconsistencies (in the location of ties) with either version of structural balance can occur

anywhere in a data structure and depend also on the details of the identified block structure, the identification of optimal signed blockmodels is not a trivial problem. The revised relocation algorithm is described in the next section, while the new repetitive branch-and-bound algorithm is described in the section following our description of the revised relocation algorithm. Both algorithms perform as intended in the sense of complementing each other fully and delineating the identical optimal partitions of a signed network. However, there are subtle differences affecting the performance of these algorithms in terms of computational time that depend on differences in the network data structures.

A Relocation Algorithm

If a network cannot be partitioned exactly according to the structure theorems, then there will be some positive ties between plus-sets and/or negative ties within plus-sets. Let $\mathcal{N}$ denote the sum of negative ties in otherwise positive blocks and $\mathcal{P}$ the sum of positive ties in negative blocks (ties between plus-sets[6]). A simple measure of the departure of some partition from the nearest ideal structure is $(\mathcal{N} + \mathcal{P})$. Letting $\mathcal{C}$ denote a clustering of the vertices and $P(\mathcal{C})$ an overall measure of departure from K balance, called a *criterion function,* then a slightly more general measure, where $0 \leq \alpha \leq 1$ is

$$P(\mathcal{C}) = \alpha\mathcal{N} + (1 - \alpha)\mathcal{P}. \tag{1}$$

If $\alpha = 0.5$, then the two types of inconsistencies are weighed the same regardless of their type. For $(0 \leq \alpha < 0.5)$, positive inconsistencies are more important, and when $(0.5 < \alpha \leq 1)$, the negative inconsistencies are considered as more important.

Seeking a partition of the vertices into plus-sets (and a partition of the arcs into blocks) is then set up as a clustering problem: Determine a clustering $\mathcal{C}^*$ such that $P(\mathcal{C}^*) = \min P(\mathcal{C})$, over all feasible clusterings of $\mathcal{V}$. By definition, $P(\mathcal{C}) \geq 0$ and equals 0 if and only if $\mathcal{N}$ and $\mathcal{P}$ are both zero. A relocation method is then employed:

1. Select a value of K.
2. Randomly determine an initial clustering, $\mathcal{C}$, with K clusters.
3. Repeat: If, in the neighborhood of the current clustering $\mathcal{C}$, there exists a clustering $\mathcal{C}'$ such that $P(\mathcal{C}') < P(\mathcal{C})$, then move to $\mathcal{C}'$. The neighborhood of a clustering $\mathcal{C}$ is determined by two transformations: (a) moving a vertex from one cluster to another cluster and (b)

interchanging two units between different clusters. This process is repeated many times (in the order of many 1,000s) to minimize the risk of reaching only a local minimum rather than a global minimum.

4. Repeat the whole procedure for different values of K.

For structural balance, determining K is made easier by the following bounds: (a) if $K = 1$, then all negative ties are inconsistent with a balanced partition, and (b) if $K = m$, all positive ties are inconsistent with perfect balance. Between these extremes, the number of inconsistencies in terms of K has a distinctive pattern for structural balance. For K, ($1 \leq K \leq m$), partitions with K and $K + 1$ plus-sets are said to be adjacent. Doreian et al. (2005) established a useful result for binary networks. For any signed network, (G, σ), there will be a unique lowest value, denoted by $P(\mathcal{C}^{\min})$, of the criterion function that occurs for partitions with a single number, K, of plus-sets or for adjacent partitions.[7] This implies that while a unique minimized value of the criterion function can be obtained by searching partitions in terms of K, it is not necessary to examine partitions for every value of K. This does not imply that there is a unique partition with the minimized value of $P(\mathcal{C}^{\min})$. Doreian and Mrvar (2009) prove that this ''nice'' property, unfortunately, does *not* hold for relaxed structural balance and that the criterion function, $P(\mathcal{C})$, declines monotonically[8] with the number of clusters, K.

Despite the evident utility of the relocation algorithm for partitioning signed networks, some problems merit further attention. These include the following:

1. The algorithm, despite the use of many repetitions, does not *guarantee* that a global minimum will be reached. Indeed, a persistent response to the use of a local optimization procedure is something like ''Yes, but you do not know if you have found a globally optimal solution.'' For small networks ($m \leq 12$), exhaustive searches are possible and confirm the solutions obtained with the heuristic relocation algorithm. So, the absence of a guarantee is not a serious problem with *very small* networks. But for larger signed networks, this remains a problem.
2. Some signed networks have multiple equally well fitting partitions for a given number of plus-sets, which can make it impossible to select any of them on the basis of the criterion function alone. (If such partitions differ only with regard to the location of a single vertex[9] and the signed blockmodels have the same block structure, then this is not a serious problem.) Of course, this problem is really not a problem with the algorithm per se and has more to do with the structure

of a given network. Nevertheless, it has an important implication that leads to a third problem.

3. Given the lack of guarantee (Item 1, above), it is possible that after using the relocation algorithm we do not have *all* of the equally well fitting partitions. As a working rule of thumb, it seems preferable to discard all of the partitions for a given value of K if there are many of them. If many equally good partitions are identified, then discarding them all is not a problem for subsequent interpretations of the network. However, if the relocation algorithm, even after many repetitions, identifies one optimal partition for a network, Item 1 coupled with Item 2 implies that there *may* be others. If so, the apparent identification of a unique optimal partition is spurious. It would be good if a guarantee could be provided both for identifying optimal blockmodel structures and for ensuring that apparent unique partitions are unique. We show that a branch-and-bound algorithm provides one.

A Repetitive Branch-and-Bound Algorithm

We present an exact algorithm for finding a K cluster partition of vertices that maximizes concordance with relaxed structural balance. This algorithm builds on a long history of branch-and-bound procedures for clustering problems (Brusco 2003, 2006; Diehr 1985; Klein and Aronson 1991; Koontz, Narendra, and Fukunaga 1975). Brusco and Steinley (forthcoming) recently devised a branch-and-bound method for generalized structural balance partitioning; however, the adaptation of their procedure is not straightforward because of differences in the assumptions regarding ideal block structure. More specifically, for structural balance, as stated earlier, the ideal block structure is known: All main diagonal blocks are positive, and all off-diagonal blocks are negative. However, for relaxed structural balance, the placement of positive and negative blocks is unknown. Therefore, the branch-and-bound procedure for relaxed structural balance partitioning must enable the determination of the ideal block structure as part of the clustering process. This is a formidable challenge because it is difficult to preserve good bounds during the solution process. To address this challenge, we employ a ''repetitive'' branch-and-bound algorithm similar in design to those developed by Brusco (2003, 2006). The repetitive branch-and-bound algorithm obtains optimal partitions for submatrices of the network matrix to establish strong bounds for the full network matrix. Although this appears to be unnecessarily time

consuming at face value, the improved bounds actually lead to a more rapid solution of larger and more difficult test problems.

Notation and Mathematical Model

$\mathcal{V}$: a set of m vertices corresponding to subjects, actors, etc., $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$;
C: the set of indices corresponding to the vertices in $\mathcal{V}$, $C = \{1, 2, \ldots, m\}$;
A: an $m \times m$ matrix of arc (or edge) weights for a signed digraph corresponding to ties among the set of m vertices;
K: the number of clusters;
$\Omega(k, l)$: the sum of positive elements in the block defined by clusters k and l, for $1 \leq k \leq K$ and $1 \leq l \leq K$;
$\Phi(k, l)$: the sum of negative elements in the block defined by clusters k and l, for $1 \leq k \leq K$ and $1 \leq l \leq K$; and
T: a $K \times K$ image matrix that characterizes the block structure of the relaxed structural balance solution, where $t_{kl} = 1$ for positive blocks and $t_{kl} = -1$ for negative blocks for $1 \leq k \leq K$ and $1 \leq l \leq K$.

Using these definitions, the optimization problem associated with relaxed structural balance partitioning can be described as follows:

$$\text{Maximize: } Z = \sum_{k=1}^{K} \sum_{l=1}^{K} \max\{\Omega(k,l), \Phi(k,l)\}, \tag{2}$$

subject to

$$\Omega(k,l) = \sum_{i \in C_k} \sum_{j \in C_l} (a_{ij} : a_{ij} > 0) \ \forall \ 1 \leq k, l \leq K; \tag{3}$$

$$\Phi(k,l) = \sum_{i \in C_k} \sum_{j \in C_l} (-a_{ij} : a_{ij} < 0) \ \forall \ 1 \leq k, l \leq K; \tag{4}$$

$$C = C_1 \cup \ldots \cup C_K; \tag{5}$$

$$|C_k| \geq 1 \quad \forall \ 1 \leq k \leq K; \tag{6}$$

$$C_k \cap C_l = \varnothing \quad \forall \ 1 \leq k < l \leq K. \tag{7}$$

The objective function value, Z, of the optimization problem represents the total concordance with relaxed structural balance. For each block, if $\Omega(k, l) \geq \Phi(k, l)$, then the block is deemed ''positive'' and $\Omega(k, l)$ is captured in the objective function while $\Phi(k, l)$ is lost because this is the absolute value of the sum of negative elements in the positive block. Similarly, if $\Omega(k, l) < \Phi(k, l)$, then the block is deemed ''negative'' and $\Phi(k, l)$ is captured in the objective function while $\Omega(k, l)$ is lost because this is the sum of positive elements in the negative block. An upper bound on the objective function, achieved if all positive elements in **A** occur in positive blocks and all negative elements in **A** occur in negative blocks, is

$$Z_{UB} = \sum_{i=1}^{n} \sum_{j=1}^{n} \left| a_{ij} \right|. \tag{8}$$

An alternative statement for achieving the upper bound is that Z_{UB} is realized if either $\Omega(k, l)$ or $\Phi(k, l)$, or both, are zero for all $1 \leq k \leq K$ and $1 \leq l \leq K$.

The image matrix, **T**, is recovered from the optimal solution to the optimization problem, which is denoted as $\{C_1^*, ..., C_K^*\}$. Defining $\Omega^*(k, l)$ and $\Phi^*(k, l)$ as the positive and negative sums, respectively, for the block formed by optimal clusters k and l, the image matrix is obtained as follows:

$$t_{kl} = \begin{cases} 1, \text{if } \Omega^*(k,l) \geq \Phi^*(k,l) \\ -1, \text{otherwise} \end{cases} \quad \forall \; 1 \leq k, l \leq K. \tag{9}$$

Implicit Enumeration and the Steps of the Branch-and-Bound Algorithm

The number of partitions of m vertices into K clusters is a Stirling number of the second kind and can be computed by using the following equation (Hand 1981):

$$\pi = \frac{1}{K!} \sum_{k=0}^{K} (-1)^k \binom{K}{k} (K-k)^m. \tag{10}$$

As noted by Brusco and Stahl (2005: chap. 2), complete enumeration of all feasible partitions is computationally feasible for only modest values of m and K. For example, there are more than 4 trillion ways to partition $m = 20$ vertices into $K = 6$ clusters, thus precluding exhaustive enumeration in a reasonable amount of microcomputer computation time.

Although branch-and-bound algorithms can require exhaustive enumeration in worst-case scenarios, they are often able to substantially reduce computation time by *implicitly* eliminating a large number of solutions from the search process. Our branch-and-bound algorithm uses a forward-search procedure that *branches* by assigning vertices to clusters. A *complete solution* in the search process corresponds to an assignment of all vertices to clusters, whereas a *partial solution* is characterized by an assignment of fewer than the total number of vertices to clusters. Through the establishment of *bounds,* we are often able to determine that a partial solution cannot possibly lead to a solution that is better than the current incumbent solution. In such instances, the partial solution is pruned and no further branches will stem from that solution. Effectively, all complete partitions that could have stemmed from the partial solution are implicitly eliminated by the pruning operation. When these pruning operations occur after assignment of only a few of the vertices, the computational savings is enormous. Each time the branch-and-bound algorithm yields a complete solution during the search process, that solution becomes the new incumbent solution. On termination of the algorithm, all partitions have been either explicitly or implicitly evaluated and the incumbent solution is a global optimum.

To preserve good bounds during the solution process, we use a repetitive approach (Brusco 2003, 2006). We begin by first applying the branch-and-bound procedure to the submatrix associated with the rows and columns of the last $K + 1$ vertices of the network matrix. Subsequently, optimal solutions are obtained for the last $K + 2$, $K + 3$, and so forth, vertices of the network matrix, culminating in the solution for the full $m \times m$ matrix. Thus, an optimal K-cluster partition is obtained for all submatrices on the interval $K + 1 \leq m' \leq m$. The solution for each submatrix is used in the bounding process of larger submatrices as necessary. For example, consider a network matrix of size $m = 30$ with $K = 5$. Assume that the optimal partitions for submatrices of size $6 \leq m' \leq 29$ have been obtained, and we are currently working on the solution for the full 30×30 matrix. Suppose that we have assigned the first $j = 5$ vertices to clusters and, therefore, know the precise objective function contribution of these 5 vertices among themselves. This is one component of the bound for this partial solution. Because we have already solved the 25×25 submatrix for the last 25 vertices in the network matrix, we also know the *best possible* contribution to the bound that could be realized among the last 25 vertices, and this is the second component of the bound. The third component of the bound is the best possible contribution to the objective function that could be realized *between* the first 5 (assigned) and last 25 (unassigned) vertices of the network matrix.

A visual description of the branch-and-bound process is displayed in Figure 1, and terminology consistent with that of Klein and Aronson

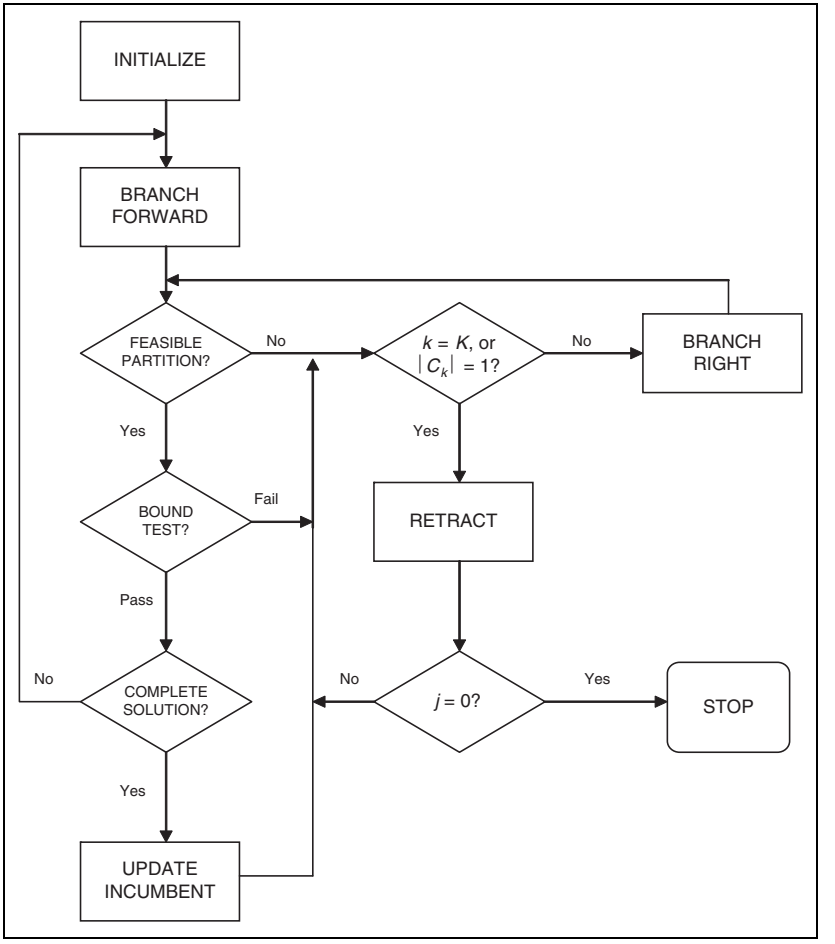

Figure 1. A flowchart of the steps of the branch-and-bound process

(1991) is used to highlight the steps. Again, it is important to recognize that the algorithm in Figure 1 is repeated $m - K$ times, each time increasing the submatrix size by one, and the final repetition obtains the solution for the full network matrix. For ease of interpretation, we describe the steps of the algorithm within the context of the final repetition.

Step 0. INITIALIZE. Select the number of clusters, *K,* and use 20 restarts of the relocation algorithm to establish an initial lower

bound, Z_{LB}, on the objective function. Set $j = 0$, $Z = 0$, $Z' = 0$, $\lambda = K$, $C_k = \varnothing$ for $1 \leq k \leq K$, $\Omega(k, l) = 0$ for $1 \leq k, l \leq K$, and $\Phi(k, l) = 0$ for $1 \leq k, l \leq K$.

Step 1. BRANCH FORWARD. Set $j = j + 1$, $k = 1$, and $C_k = C_k \cup \{j\}$. If $|C_k| = 1$, then set $\lambda = \lambda - 1$. Next, perform the following substeps:

Step 1a. Set $\Omega(k,l) = \Omega(k,l) + \max\{a_{ji}, 0\}, \forall\, 1 \leq l \neq k \leq K$ and $i \in C_l$,

Step 1b. Set $\Omega(l,k) = \Omega(l,k) + \max\{a_{ij}, 0\}, \forall\, 1 \leq l \neq k \leq K$ and $i \in C_l$,

Step 1c. Set $\Phi(k,l) = \Phi(k,l) + \left|\min\{a_{ji}, 0\}\right|, \forall\, 1 \leq l \neq k \leq K$ and $i \in C_l$,

Step 1d. Set $\Phi(l,k) = \Phi(l,k) + \left|\min\{a_{ij}, 0\}\right|, \forall\, 1 \leq l \neq k \leq K$ and $i \in C_l$.

Step 2. FEASIBLE PARTITION? If $m - j < \lambda$, then go to Step 6.

Step 3. BOUND TEST. Perform the following substeps:

Step 3a. Set $Z = \sum_{k'=1}^{K} \sum_{l=1}^{K} \max\{\Omega(k', l), \Phi(k', l)\}$ as the direct contribution to the objective function from the assigned vertices.

Step 3b. Retrieve $Z^*(m - j)$ as the optimal objective function value for the yet unassigned $m - j$ vertices. This value, which was obtained from an earlier repetition of the branch-and-bound algorithm, provides an upper bound on the best possible objective value contribution that could be realized from among the unassigned vertices.

Step 3c. Compute Z' as an upper bound on the contribution that can occur between the j assigned vertices and the $m - j$ unassigned vertices. This bound is computed as follows:

$$\Gamma_1(i,k) = \sum_{h=1}^{j} (\max\{a_{hi}, 0\} | h \in C_k), \text{for } j + 1 \leq i \leq m;$$

$$\Gamma_2(i,k) = \sum_{h=1}^{j} (\max\{a_{ih}, 0\} | h \in C_k), \text{for } j + 1 \leq i \leq m;$$

$$\theta_1(i,k) = \sum_{h=1}^{j} (\max\{-a_{hi}, 0\} | h \in C_k), \text{for } j + 1 \leq i \leq m;$$

$$\theta_2(i,k) = \sum_{h=1}^{j} (\max\{-a_{ih}, 0\} | h \in C_k), \text{ for } j+1 \le i \le m;$$

$$Z' = \sum_{i=j+1}^{e} \sum_{k'=1}^{K} (\max\{\Gamma_1(i,k'), \theta_1(i,k')\} + \max\{\Gamma_2(i,k'), \theta_2(i,k')\}).$$

Step 3d. If $(Z + Z^*(m-j) + Z') \le Z_{LB}$, then go to Step 6.
Step 4. COMPLETE SOLUTION? If $j < m$, then go to Step 1.
Step 5. UPDATE INCUMBENT. Set $Z_{LB} = Z$ and $C_l^* = C_l$ for $1 \le l \le K$.
Step 6. ALLOCATION. If $k = K$ or $|C_k| = 1$, then go to Step 8; otherwise proceed to Step 7.
Step 7. BRANCH RIGHT. Perform the following substeps:
Step 7a. Set $\Omega(k,l) = \Omega(k,l) - \max\{a_{ji}, 0\}, \forall\, 1 \le l \ne k \le K$ and $i \in C_l$,
Step 7b. Set $\Omega(l,k) = \Omega(l,k) - \max\{a_{ij}, 0\}, \forall\, 1 \le l \ne k \le K$ and $i \in C_l$,
Step 7c. Set $\Phi(k,l) = \Phi(k,l) - |\min\{a_{ji}, 0\}|, \forall\, 1 \le l \ne k \le K$ and $i \in C_l$,
Step 7d. Set $\Phi(l,k) = \Phi(l,k) - |\min\{a_{ij}, 0\}|, \forall\, 1 \le l \ne k \le K$ and $i \in C_l$,
Step 7e. Set $C_k = C_k - \{j\}$, $k = k+1$, $C_k = C_k \cup \{j\}$.
Step 7f. If $|C_k| = 1$, then set $\lambda = \lambda - 1$.
Step 7g. Set $\Omega(k,l) = \Omega(k,l) + \max\{a_{ji}, 0\}, \forall\, 1 \le l \ne k \le K$ and $i \in C_l$,
Step 7h. Set $\Omega(l,k) = \Omega(l,k) + \max\{a_{ij}, 0\}, \forall\, 1 \le l \ne k \le K$ and $i \in C_l$,
Step 7i. Set $\Phi(k,l) = \Phi(k,l) + |\min\{a_{ji}, 0\}|, \forall\, 1 \le l \ne k \le K$ and $i \in C_l$,
Step 7j. Set $\Phi(l,k) = \Phi(l,k) + |\min\{a_{ij}, 0\}|, \forall\, 1 \le l \ne k \le K$ and $i \in C_l$.
Step 7k. Go to Step 2.
Step 8. RETRACT. Perform the following substeps.
Step 8a. Set $\Omega(k,l) = \Omega(k,l) - \max\{a_{ji}, 0\}, \forall\, 1 \le l \ne k \le K$ and $i \in C_l$,
Step 8b. Set $\Omega(l,k) = \Omega(l,k) - \max\{a_{ij}, 0\}, \forall\, 1 \le l \ne k \le K$ and $i \in C_l$,
Step 8c. Set $\Phi(k,l) = \Phi(k,l) - |\min\{a_{ji}, 0\}|, \forall\, 1 \le l \ne k \le K$ and $i \in C_l$,

Step 8d. Set $\Phi(l,k) = \Phi(l,k) - \left|\min\{a_{ij}, 0\}\right|, \forall\, 1 \leq l \neq k \leq K$ and $i \in C_l$,
Step 8e. Set $C_k = C_k - \{j\}$.
Step 8f. If $|C_k| = \varnothing$, then set $\lambda = \lambda + 1$.
Step 8g. Set $j = j - 1$.
Step 8h. If $j = 0$, then STOP; otherwise, set $k = l : j \in C_l$ and go to Step 6.

The cluster assignments and bounds are initialized in Step 0. The vertex index pointer, *j,* is advanced in Step 1, and the vertex index is assigned to cluster $k = 1$. Steps 1a through 1d augment the contribution to the positive and negative block sums from the actual assignment of j to cluster k. If $m - j < \lambda$ at Step 2, then there are too few vertex indices remaining to fill the empty clusters, which results in control being passed to Step 6 for allocation of the partial solution. If $m - j \geq \lambda$ at Step 2, then the bound test is performed in Step 3. The first component of the upper bound on the objective function that can be realized via completion of the partial solution is the objective function value of the current partial solution (Z) computed in Step 3a. The second component from Step 3b is an upper bound on the contribution among the $m - j$ unassigned vertices, which is obtained from the branch-and-bound algorithm on an earlier repetition. The third component from Step 3c is computed as an upper bound on the contribution that can occur between the assigned and unassigned vertices. If $(Z + Z^*(m - j) + Z')$ is less than or equal to Z_{LB}, then pursuing the current partial solution cannot possibly lead to a complete solution with a better objective function value than Z_{LB} and control is passed to Step 6 for allocation. If a partial solution reaches Step 4, then it has passed the pruning tests and a check is made to see if the solution is, in fact, complete. If $j < m$ at Step 4, then control is passed to Step 1 for advancement of the vertex index pointer and assignment; however, if $j = m$, then control is passed to Step 5 where the complete solution becomes the new incumbent partition and provides a new lower bound, Z_{LB}.

Step 6 allocates a solution for either a ''branch right'' operation in Step 7 or retraction in Step 8. If $k < K$ and $|C_k| > 1$ in Step 6, then control is passed to Step 7, which performs the branching operation by moving the current vertex from cluster k to cluster $k + 1$ (i.e., the cluster on the right). Steps 7a through 7d remove the contribution stemming from the assignment of vertex j to its current cluster k. Step 7e moves the vertex to the next cluster, and Step 7f reduces the number of empty clusters if j is the first vertex in that new cluster. The contribution to the objective function for vertex j in its new cluster is updated in Steps 7g through 7j. Control is passed back to Step 2 via Step

7k. If $k = K$ or $|C_k| = 1$ in Step 6, then there is no reason to move the current vertex index to the next cluster and control is passed to Step 8 for retraction. Retraction involves traversing backward in the search tree, and when this traversal returns the algorithm to the root node of the branch-and-bound search tree (i.e., $j = 0$), the algorithm terminates.

Modified Branch-and-Bound Algorithm for Finding Multiple Optima

On convergence, the branch-and-bound algorithm finds a partition of vertices that is guaranteed to be a globally optimal solution to the partitioning problem posed by equations (2) through (7). The obtained partition, however, is not necessarily the only globally optimal partition. We have developed a modified version of the branch-and-bound algorithm that will store up to 5,000 globally optimal partitions, which is more than sufficient to accommodate all of the optimal partitions in typical test problems from the literature. The program can be modified to store more than 5,000 partitions, if necessary; however, limits on computer memory can present a problem if the number of multiple optima far exceeds 5,000. We recommend that the branch-and-bound algorithm be implemented to obtain the optimal objective function value, Z^*. Next, we use $Z_{LB} = Z^*$ in the modified program to find all partitions that yield the optimal objective function value. The modified algorithm required only minor adjustments to Steps 4 and 6 of the algorithm. In Step 4, we replace $\leq$ with $<$ to ensure that partitions with the optimal objective function value will be collected, and a routine is added in Step 5 to enumerate and store the optimal partitions.

As noted earlier, multiple optima have important pragmatic ramifications. When alternate optima differ only by relocation of one or two vertices across clusters, the block structures are the same. When two different block structures yield the same objective function, this suggests instability in the partitioning process. As with the relocation method, values of K for which multiple block structures exist are typically discarded.

Comparing the Objectives of the Branch-and-Bound and Relocation Algorithms

At first sight, the branch-and-bound and relocation algorithms appear to differ and raise the issue for finding different outcomes. However, this is not the case. Consider the term $Z = \sum_{k=1}^{K} \sum_{l=1}^{K} \max\{\Omega(k,l), \Phi(k,l)\}$ from the

branch-and-bound algorithm. If we accept the identification of positive and negative blocks, then maximizing Z while summing over the blocks is equivalent to minimizing $P(\mathcal{C})$. Given the presentation of the relocation algorithm, let $\mathcal{P}^{\mathrm{T}}$ be the sum of the values for all of the positive ties in the network and let $\mathcal{N}^{\mathrm{T}}$ be the sum for all of the negative ties in the network. The sum of the correctly located positive ties relative to structural balance is $(\mathcal{P}^{\mathrm{T}} - \mathcal{P})$, and the sum of the correctly located negative ties relative to structural balance is $(\mathcal{N}^{\mathrm{T}} - \mathcal{N})$. Maximizing Z is equivalent to maximizing $[(\mathcal{P}^{\mathrm{T}} - \mathcal{P}) + (\mathcal{N}^{\mathrm{T}} - \mathcal{N})]$, and this is the same as maximizing $[(\mathcal{P}^{\mathrm{T}} + \mathcal{N}^{\mathrm{T}}) - (\mathcal{P} + \mathcal{N})]$. Given that $(\mathcal{P}^{\mathrm{T}} + \mathcal{N}^{\mathrm{T}})$ is fixed for a network, this maximization is equivalent to minimizing $(\mathcal{P} + \mathcal{N})$. The same argument[10] holds when differential weighting of the positive and negative ties is included when using α in equation (1). Under the assumption of $\alpha = 0.5$, the criterion function in equation (1) can be computed for the branch-and-bound solution as follows:

$$P(X) = \frac{(Z_{UB} - Z^{*})}{2}. \tag{11}$$

Results and Analyses for Empirical Network Matrices

Modestly Sized Network Matrices

We applied the branch-and-bound procedure to five network matrices from the blockmodeling literature. The first three of these matrices correspond to data obtained from female students at three different off-campus dormitories at an eastern college. The data were collected by Lemann and Solomon (1952). Blockmodeling analyses of these matrices were performed by Doreian (2008). The three matrices are labeled House A ($m = 21$), House B ($m = 18$), and House C ($m = 20$). The fourth matrix was obtained from a well-known sociometric study originally conducted by Sampson (1968), with subsequent analyses by numerous authors (Breiger et al. 1975; Doreian 2008; Doreian and Mrvar 1996, 2009; Faust 1988; White, Boorman, and Breiger 1976). Following Doreian and Mrvar (2009), we summed the ''affect,'' ''esteem,'' and ''respect'' relations at time period four to obtain an 18×18 matrix of signed relations. We refer to this matrix as ''Sampson_T4.'' The fifth data set also corresponds to a well-known sociometric study of $m = 17$ men at a university dormitory (Newcomb 1961). We consider a particular network matrix produced from the original data by Doreian and Mrvar (2009). One of the two optimal 4-cluster partitions for these data is shown in Figure 2 to give a sense of the partition structures produced. The squares and diamonds in Figure 2 represent positive and negative

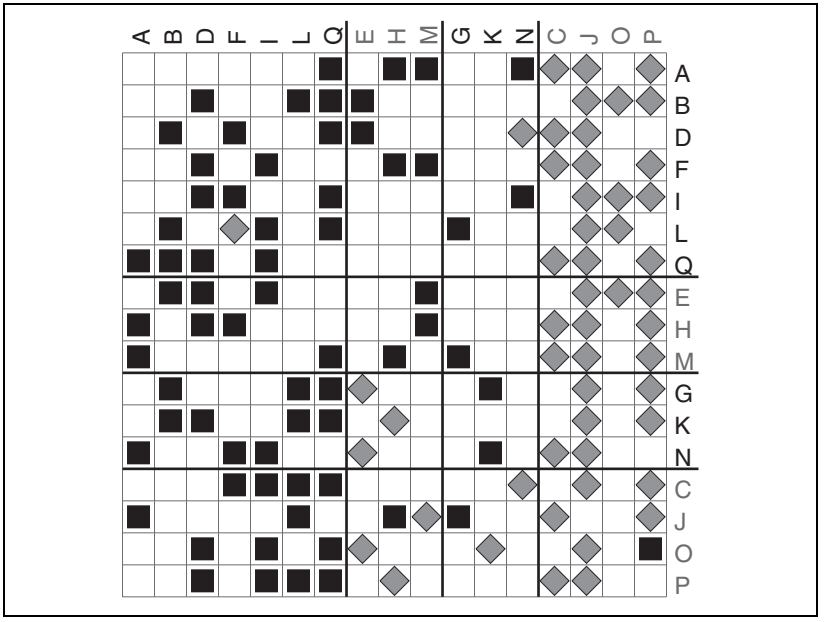

Figure 2. Partition structure for the Newcomb final week: $K = 4$
Note: The squares represent positive ties, and the diamonds represent negative ties.

ties, respectively. There are five inconsistencies with relaxed structural balance in Figure 2, resulting in a criterion function value from equation (1) of 2.5 with $\alpha = 0.5$.

For each of these matrices, we applied the branch-and-bound and relocation algorithms six times, once each for $2 \leq K \leq 7$ clusters, resulting in a total of 30 test structures. For each test, data were collected with respect to the upper bound (Z_{UB}), the optimal objective function value, Z^*, and the criterion function in (1) assuming $\alpha = 0.5$. Next, using the obtained value of Z^* as input, we ran the modified version of the branch-and-bound program that obtains *all* of the multiple optima. The exact number of unique optima was collected for every analysis. Table 1 provides the results for each of the five network matrices at each of the six values of K.

The salient results of our computational results are for each test structure: (i) the branch-and-bound algorithm obtained and confirmed a globally optimal partition, (ii) the relocation algorithm obtained a globally optimal partition, (iii) the modified branch-and-bound algorithm provided all of the

Table 1. Computational Results for Empirical Network Matrices

Data set	K	Z_{UB}	Z	Penalty index criterion function	Number of optima
House (A)	2	504	408	48.0	1
	3	504	454	25.0	1
	4	504	473	15.5	1
	5	504	477	13.5	9
	6	504	483	10.5	1
	7	504	487	8.5	1
House (B)	2	406	322	42.0	1
	3	406	337	34.5	1
	4	406	350	28.0	2
	5	406	363	21.5	1
	6	406	373	16.5	2
	7	406	378	14.0	1
House (C)	2	462	398	32.0	1
	3	462	409	26.5	2
	4	462	419	21.5	4
	5	462	427	17.5	6
	6	462	433	14.5	2
	7	462	438	12.0	4
Sampson (Time 4)	2	612	526	43.0	1
	3	612	558	27.0	1
	4	612	576	18.0	1
	5	612	587	12.5	1
	6	612	596	8.0	1
	7	612	600	6.0	4
Newcomb (Week 15)	2	119	109	5.0	2
	3	119	112	3.5	5
	4	119	114	2.5	2
	5	119	116	1.5	3
	6	119	118	0.5	2
	7	119	119	0.0	5

globally optimal partitions, and (iv) the relocation algorithm identified all of the globally optimal partitions. At a minimum, use of the branch-and-bound algorithm, with its guarantee of locating genuinely optimal solutions, appears to settle the issue regarding the argument that the relocation algorithm need not identify optimal partitions for most networks of this size in favor of this method. Even so, it would be prudent to not assume that the relocation method will always do this because the performances of algorithms are affected

by network structures. To explore this issue further, we examine a larger empirical network matrix.

A Larger, More Challenging Empirical Matrix

The identification of large signed network matrices that are published in the social network literature is a difficult task. After some searching, we were able to locate a large signed network based on data collected by McKinney (1948), who sought to discover what relationships existed among children in a ninth-grade classroom as well as their expressed reasons for the choices they made. They were ''asked to express their attitude towards serving in a discussion group with other members of the class'' (McKinney 1948:357).[11] The data were recorded as being willing to serve with other children (+1), not being willing to serve (–1), and indifferent (0). We extracted the reciprocated positive and negative preferences for participating with others. These relational data are much denser than the signed data usually collected in this early era of sociometric data collection when choices, both positive and negative, were restricted in the prevailing fixed choice instruments (usually three choices).

The McKinney network matrix presents a more formidable challenge than those analyzed in the previous subsection, for three reasons: (1) It is a larger (29×29) network, (2) the relational density is high (31.4 percent), and (3) it has a skewed distribution of nonzero matrix elements (246 elements of 1, but only 18 elements of –1). We applied the branch-and-bound algorithm to the McKinney network matrix for 2, 3, and 4 clusters. In each instance, an optimal partition was obtained in less than five seconds. The optimal criterion function values for 2, 3, and 4 clusters were 4.0, 1.0, and 0.0, respectively. Unfortunately, the optimal partitions were not unique. In fact, we generated 30, 873, and 3,528 unique optimal partitions for 2, 3, and 4 clusters, respectively. One of the optimal 4-cluster partitions is shown in Figure 3.

The relocation heuristic experienced considerable difficulty with the McKinney network matrix. For $K = 4$ clusters, we restarted the heuristic five times, with 1 million repetitions for each restart. Each restart required approximately five minutes of computation time, and the best-found criterion function was 1.0. Accordingly, for this particular network matrix, there was considerable value afforded by the branch-and-bound algorithm with respect to both finding an optimal partition and generating a large set of alternative optimal solutions. Our conjecture is that the principal reason that the relocation heuristic struggled with this particular matrix is the tremendous disparity in the number of positive and negative elements.

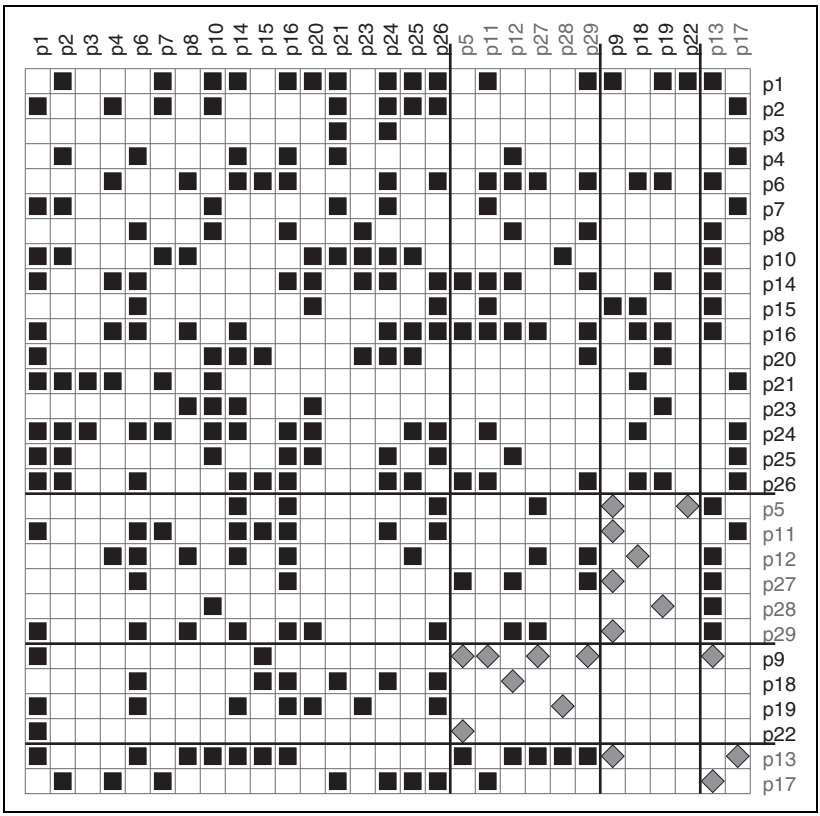

Figure 3. Partition structure for the McKinney data: $K = 4$

Note: The squares represent positive ties, and the diamonds represent negative ties.

Simulation Experiment

Test Problems

Most of the empirical signed network matrices in the blockmodeling literature are of size 25×25 or smaller. The branch-and-bound algorithm typically obtains optimal partitions for most networks of this size within a few minutes of microcomputer computation time, and often within seconds. Moreover, as noted above, the relocation algorithm has no difficulty locating the optimal partition for empirical networks of this size. To provide a more in-depth investigation of the performances of the branch-and-bound and

relocation algorithms, we conducted a simulation experiment. Test problems were generated by manipulating five network features, each with two levels. The first feature, the number of vertices, was set at levels of $m = 20$ and $m = 40$. The levels of the second feature, the number of clusters for the vertices, were $K = 3$ and $K = 5$. The third feature, within-block density, represents the approximate percentage of nonzero elements within each block of the network matrix. The levels of the third feature were 40 percent and 80 percent. The levels of the fourth feature, the percentage of positive blocks, were 50 percent and 70 percent. For example, given that the total number of blocks is K^2, the 70 percent setting would imply that the number of positive blocks is $\lceil .7 \times K^2 \rceil$ and the number of negative blocks is $K^2 - \lceil .7 \times K^2 \rceil$, where $\lceil g \rceil$ denotes the smallest integer greater than or equal to g. The fifth feature, the level of error perturbation, was tested at levels of $\varepsilon = 10$ percent and $\varepsilon = 20$ percent.

The generation of a test problem begins with the assignment of the m vertices into K clusters, such that the number of vertices in each cluster is approximately the same. Next, either 50 percent or 70 percent of the K^2 blocks were specified as positive, and the remaining blocks were negative. The locations of the positive blocks in the image matrix were randomly selected. All elements in the network matrix were initially assigned a value of 0. Within each positive (negative) block, each element within the block had either a 40 percent chance or an 80 percent chance of being changed from 0 to +1 (–1); however, all diagonal elements remained equal to 0. Upon completion of the process, the network matrix has a ''perfect'' relaxed structural balance because all positive blocks have elements that are either 0 or 1 and all negative blocks have elements that are either 0 or –1. To eliminate this perfect structure, the sign of each nonzero matrix element had either a 10 percent or 20 percent chance of being reversed. We found that this perturbation process maintained sufficient structure in the network, while at the same time providing two distinct degrees of departure from perfect structure.

The manipulation of five features at two levels each resulted in an experimental design with $2^5 = 32$ cells. Three test problem replicates were generated for each cell, resulting in a total of 96 unique test problems.

Performance Measures

The simulation experiment was designed to answer three questions: (1) What problem characteristics result in greater difficulty for the branch-and-bound algorithm with respect to producing optimal solutions? (2) What problem characteristics result in greater difficulty for the relocation algorithm with

respect to locating optimal solutions? and (3) Are the problem characteristics that make reaching solutions more difficult for the branch-and-bound algorithm necessarily the same as those that create difficulty for the relocation algorithm in finding an optimal solution?

In light of the fact that the branch-and-bound algorithm does not use multiple restarts, the best measure of problem difficulty is total computation time. For this reason, the principal performance metric for the branch-and-bound algorithm is the total time required to obtain and confirm the globally optimal solution. In contrast, the relocation heuristic does use multiple restarts and computation time is primarily affected by the choice of the number of restarts. It is important to remember, however, that even for a large number of restarts, the relocation heuristic is quite efficient, with far less computation time variability than the branch-and-bound algorithm. For example, the branch-and-bound algorithm might solve one 40-vertex problem in seconds but require multiple hours for a different 40-vertex problem. For the same two problems, application of the relocation heuristic with the same number of restarts is apt to require no more than a few seconds in both instances.

We use the ''attraction rate'' as measure of performance for the relocation heuristic. The attraction rate is the percentage of restarts for which the globally optimal solution was obtained. For example, if we ran the relocation heuristic using 10,000 restarts and obtained the globally optimal solution 100 times, then the attraction rate would be 100/10,000 = 1 percent. Thus, our estimate of the probability of the relocation heuristic finding the globally optimal solution for any given randomly generated starting solution is 1 percent. Note that even an attraction rate of only 1 percent suggests that the relocation heuristic would almost certainly find the optimal solution in 1,000 restarts. The probability of *not* finding the optimal solution in 1,000 restarts could be estimated as $(1 - .01)^{1,000} = 4.32 \times 10^{-5}$, and 2,000 restarts reduces this probability to $(1 - .01)^{2,000} = 1.86 \times 10^{-9}$. The key here is that even small attraction rates often correspond to very high probabilities of finding the optimal solution for a modest number of restarts (e.g., 10,000).

Results

The branch-and-bound computation time and relocation heuristic attraction rate for each of the 96 test problems are provided in the appendix (which can be accessed online at http://smr.sagepub.com/supplemental). A summary of the results is provided in Table 2. For each level of each feature, Table 2 provides the average branch-and-bound computation time and average

Table 2. Simulation Results: A Summary for the Levels of Each Feature

Feature levels	Mean computation time for the branch-and-bound algorithm	Mean attraction rate for the relocation heuristic
$m = 20$ actors	85.27	47.79
$m = 40$ actors	207.68	49.88
$K = 3$ clusters	2.55	49.71
$K = 5$ clusters	290.40	47.96
40% within-cluster density	285.75	45.50
80% within-cluster density	7.21	52.17
50% positive blocks	108.86	73.48
70% positive blocks	184.09	24.19
10% error perturbation	17.53	50.17
20% error perturbation	275.42	47.50
Overall averages	146.48	48.33

relocation heuristic attraction rate. The averages across all test problems are also provided.

The computation times for the branch-and-bound algorithm ranged from less than 1 second to nearly 75 minutes, with an average of 146.48 seconds. The number of clusters, *K,* had the greatest effect on computation time. All 3-cluster problems were solved in 12 seconds or less; however, 14 of the 5-cluster problems required at least 100 seconds, and 5 of these required more than 1,000 seconds. Problem density and error perturbation also had strong effects on computation time. All 14 of the problems requiring more than 100 seconds were 40 percent block density, and 12 of these 14 problems had an error perturbation of $\varepsilon = .02$.

The relocation heuristic was run on different hardware and software platforms; however, it is fair to conclude that 100,000 restarts of the relocation heuristic required far less time than even the average branch-and-bound time. Thus, the computation time advantage for the heuristic is unequivocal. The overall attraction rate for the relocation heuristic was an astonishing 48.33 percent, revealing the general efficacy of this procedure. An inspection of the attraction rates for each feature level, which are provided in Table 2, reveals that four of the five features did not seem to have a profound effect on this performance measure. For example, the number of clusters (K), which had the strongest effect on branch-and-bound computation time, appears to have only a limited effect on the attraction rate. The average attraction rates at $K = 3$ and $K = 5$ are 49.71 percent and 47.96 percent, respectively. The

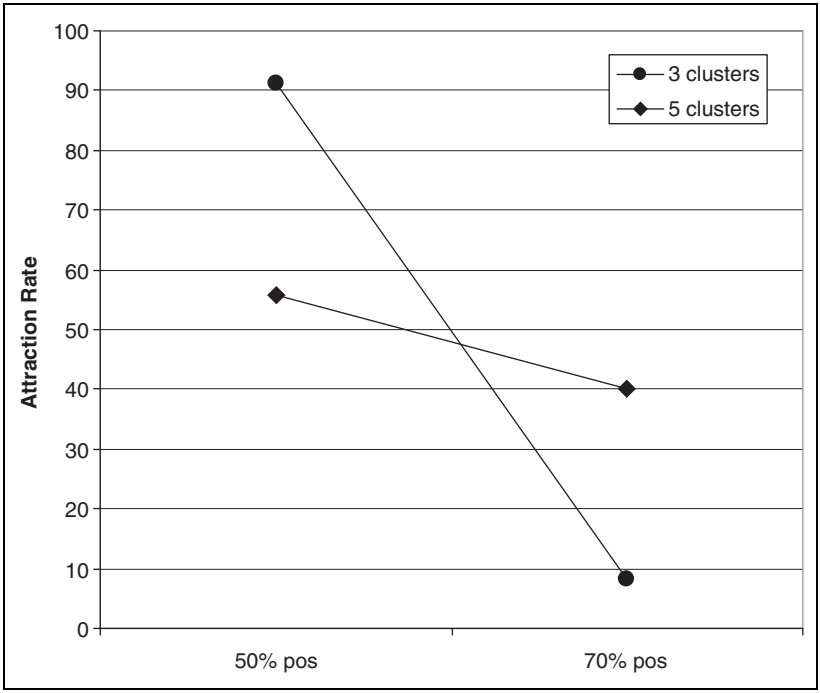

Figure 4. A Number of Clusters $\times$ Percentage of Positive Blocks interaction plot for the attraction rate of the relocation heuristic

only feature with a clearly evident strong effect on attraction rate was the percentage of positive blocks, where the average attraction rates were 73.48 percent and 24.19 percent for the levels of 50 percent and 70 percent positive blocks, respectively.

Our investigation of feature interactions on attraction rates revealed an especially interesting relationship between the number of clusters and the percentage of positive blocks. A visual description of this relationship is provided in Figure 4. At $K = 5$ clusters and 50 percent positive blocks, the average attraction rate was 55.83 percent. The average attraction rate for $K = 5$ clusters and 70 percent positive blocks is slightly lower at 40.08 percent. The attraction rate differences at $K = 3$ clusters present a much different scenario. At $K = 3$ and 50 percent positive blocks, the average attraction rate was an 91.13 percent; however, at $K = 3$ and 70 percent positive blocks, the average attraction rate was only 8.30 percent. The strong interaction with

crossover effect displayed in Figure 4 clearly reveals that the number of clusters does affect attraction rate, a finding that is masked by the approximately equal averages for $K = 3$ and $K = 5$ shown in Table 2. The largest attraction rates associated with the relocation heuristic were observed for 3-cluster problems with 50 percent positive blocks, but the smallest attraction rates were observed for 3-cluster problems with 70 percent positive blocks. In fact, there were four instances of 3-cluster, 70 percent positive-block problems where the attraction rate was less than 0.1 percent.

Together, the results in Table 2 and Figure 4 suggest that the characteristics that make problems difficult for the branch-and-bound algorithm are not the same as those that create challenges for the relocation heuristic. For example, the percentage of positive blocks had the largest effect on the attraction rate of the relocation heuristic, but this same feature had the smallest effect on the computation time of the branch-and-bound algorithm. Perhaps even more interesting are the differences observed regarding the number of clusters. The number of clusters had the strongest effect on branch-and-bound computation time, with problems at $K = 5$ taking much longer to solve than those for $K = 3$. Contrastingly, the attraction rates for the relocation heuristic were consistently large at $K = 5$ clusters but differed greatly for $K = 3$ clusters, depending on the percentage of positive blocks. In summary, 5-cluster problems were the most difficult for the branch-and-bound algorithm, whereas 3-cluster problems with 70 percent positive blocks were the most challenging for the relocation heuristic. We note that the difficulties encountered when using the relocational algorithm on the McKinney data are consistent with these simulation results.

Conclusion

Comparing and Contrasting the Two Algorithms

The branch-and-bound and relocation algorithms each possess certain advantages relative to one another. The principal, and perhaps only, advantage of the branch-and-bound algorithm can be summed up in one word: *guarantee*. The branch-and-bound algorithm provides an unequivocal guarantee of a globally optimal partition that is not afforded by the relocation heuristic. Moreover, the modified branch-and-bound algorithm can collect all of the globally optimal partitions, provided that the number of global optima would not result in storage requirements that exceed computer memory. The relocation algorithm also stores multiple optimal partitions; however, there is no guarantee that all of the optima will be identified.

The advantage of the branch-and-bound procedure notwithstanding, there are many compelling arguments for preferring the relocation procedure. First, as our results suggest, for problems that can be tackled using the branch-and-bound approach, the relocation heuristic also obtains the globally optimal partition. Second, the scalability of the branch-and-bound procedure is limited by the values of m and K, as well as the distribution of elements in the network matrix, **A**. As noted previously, if the bounds for a particular network matrix are not sharp, the branch-and-bound procedure can devolve into a near-complete enumeration of all partitions, which can result in astronomical computation times. In contrast, the relocation algorithm can accommodate large m and K, and its scalability is much less affected by the properties of the network matrix. Yet, it is not immune to these problems. Third, a critical advantage of the relocation algorithm is flexibility. Modification of the branch-and-bound algorithm to accommodate alternative objective criteria, or the identification of null blocks in addition to positive and negative blocks, is nontrivial. The relocation algorithm can be adapted easily for alternative block types and objective functions. Fourth, the relocation algorithm has the advantage of availability in the Pajek software system for social networks (see de Nooy, Mrvar, and Batagelj 2005, for a systematic presentation of Pajek).

In summary, the new branch-and-bound algorithm has inherent mathematical value as a methodological approach for obtaining guaranteed optimal partitions for problems of nontrivial size. Moreover, its development has provided a mechanism for benchmarking the performance of Doreian and Mrvar's (2009) relocation heuristic with respect to finding all of the optimal partitions for well-structured networks of modest size. In light of this contribution, we can now confidently conclude that the relocation procedure is obtaining optimal partitions to modestly sized problems from this literature, which solidifies its position as the recommended methodology for relaxed structural balance partitioning. In terms of the theory–model–data triangle, the convergence of two algorithms provides more confidence for establishing partitions of signed networks consistent with the formalization of relaxed structural balance. Yet, we caution against using only the relocation algorithm for networks larger than the empirical ones considered here. Moreover, there are design features of network structures that were not considered in our construction of synthetic test problems (e.g., matrix symmetry).

Extensions

There would seem to be considerable room for the development and evaluation of exact and approximate algorithms for relaxed structural balance.

With respect to exact procedures, the establishment of stronger pruning rules could improve the scalability of branch-and-bound methods. Alternatively, approaches based on column generation and/or cutting planes have proven effective for similar classes of clustering problems (see Hansen and Jaumard 1997) and should provide an important avenue for future research. In addition to expanding the computational feasibility of exact approaches, efforts could also be made to improve their flexibility. For example, exact methods for more general block structures, as well as two-mode proximity data, would provide a valuable contribution.

Although we have shown that the relocation heuristic obtained optimal solutions for the modestly sized networks considered in this article, these results do not necessarily generalize to larger problem instances. For example, in other partitioning contexts (Brusco and Köhn, forthcoming; Brusco and Steinley 2007), experimental studies have shown that multistart implementations of relocation algorithms can be outperformed by more sophisticated metaheuristics such as genetic algorithms, simulated annealing, tabu search, and variable neighborhood search. Accordingly, another interesting area for further investigation is the evaluation of the relocation heuristic in comparison to metaheuristic procedures.

Acknowledgements

We appreciate greatly the helpful comments of three reviewers. These comments helped us improve the manuscript in terms of its content and presentation.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interests with respect to the authorship and/or publication of this article.

Financial Disclosure/Funding

The author(s) received no financial support for the research and/or authorship of this article.

Notes

1. While the first published statement for this program appeared in 1998, it is continually updated to include more network analytic methods as new ideas are created and better algorithms are established. The same is true for UCINET.

2. Because n is used in the structural balance literature to denote negative ties, we use m to denote the number of actors (vertices) in a network.
3. Equivalently, an alternative representation is σ: $\mathcal{A} \rightarrow \{+1, -1\}$ where +1 and –1 denote the signs.
4. For formal completeness, one of these sets may be empty.
5. When partitioning signed networks that are not exactly K balanced, an identified positive block can contain negative ties as inconsistencies and identified negative blocks can contain positive ties as inconsistencies.
6. Note that if the network is a binary network then $\mathcal{P}$ is the number of the positive ties between plus-sets and $\mathcal{N}$ is the number of negative ties within plus-sets.
7. We have encountered no examples using valued signed networks that contradict this result and conjecture that it generalizes for valued signed networks.
8. This behavior is the same as for the criterion function for structural equivalence.
9. This mirrors the phenomenon of finding ''floaters'' for an early blockmodeling program proposed by Heil and White (1976) in terms of structural equivalence.
10. This assumes that the diagonal of the relational matrix, A, has only null values. Usually, this is the case for social networks.

References

Batagelj, V. and A. Mrvar. 1998. ''Pajek—Program for Large Network Analysis.'' *Connections* 21:47-57.

Borgatti, S. P., M. G. Everett, and L. Freeman. 2002. *Ucinet for Windows: Software for Social Network Analysis*. Harvard, MA: Analytic Technologies.

Breiger, R. L., S. A. Boorman, and P. Arabie. 1975. ''An Algorithm for Clustering Relational Data With Applications to Social Network Analysis and Comparison to Multidimensional Scaling.'' *Journal of Mathematical Psychology* 12:328-83.

Brusco, M. J. 2003. ''An Enhanced Branch-and-Bound Algorithm for a Partitioning Problem.'' *British Journal of Mathematical and Statistical Psychology* 56:83-92.

Brusco, M. J. 2006. ''A Repetitive Branch-and-Bound Procedure for Minimum Within-Cluster Sums of Squares Partitioning.'' *Psychometrika* 71:347-63.

Brusco, M. J. and H.-F. Köhn. Forthcoming. ''Clustering Qualitative Data Based on Binary Equivalence Relations: Neighborhood Search Heuristics for the Clique Partitioning Problem.'' *Psychometrika*.

Brusco, M. J. and S. Stahl. 2005. *Branch-and-Bound Applications in Combinatorial Data Analysis*. New York: Springer.

Brusco, M. J. and D. Steinley. 2007. ''A Comparison of Heuristic Procedures for Minimum Within-Cluster Sums of Squares Partitioning.'' *Psychometrika* 72:583-600.

Brusco, M. J. and D. Steinley. Forthcoming. ''*K*-Balance Partitioning: An Exact Method With Applications to Generalized Structural Balance and Other Psychological Contexts.'' *Psychological Methods*.

Burt, R. S. 1976.'' Positions in Networks.'' *Social Forces* 55:93-122.

Cartwright, D. and F. Harary. 1956. ''Structural Balance: A Generalization of Heider's Theory.'' *Psychological Review* 63:277-93.

Davis, J. A. 1967. ''Clustering and Structural Balance in Graphs.'' *Human Relations* 20:181-87.

de Nooy, W., A. Mrvar, and V. Batagelj. 2005. *Exploratory Social Network Analysis With Pajek*. New York: Cambridge University Press.

Diehr, G. 1985. ''Evaluation of a Branch and Bound Algorithm for Clustering.'' *SIAM Journal for Scientific and Statistical Computing* 6:268-84.

Doreian, P. 2008. ''A Multiple Indicator Approach to Blockmodeling Signed Networks.'' *Social Networks* 30:247-58.

Doreian, P., V. Batagelj, and A. Ferligoj. 2005. *Generalized Blockmodeling*. Cambridge, England: Cambridge University Press.

Doreian, P. and A. Mrvar. 1996. ''A Partitioning Approach to Structural Balance.'' *Social Networks* 18:149-68.

Doreian, P. and A. Mrvar. 2009. ''Partitioning Signed Social Networks.'' *Social Networks* 31:1-11.

Faust, K. 1988. ''Comparison of Methods for Positional Analysis: Structural and General Equivalences.'' *Social Networks* 10:313-41.

Feld, S. and R. Elmore. 1982. ''Patterns of Sociometric Choices: Patterns of Transitivity Reconsidered.'' *Social Psychology Quarterly* 45(2):77-85.

Hand, D. J. 1981. *Discrimination and Classification*. New York: John Wiley.

Hansen, P. and B. Jaumard. 1997. ''Cluster Analysis and Mathematical Programming.'' *Mathematical Programming* 79:191-215.

Harary, F. 1953. ''On the Notion of Balance in Signed Graphs.'' *Michigan Mathematical Journal* 2:143-46.

Harary, F., R. Z. Norman, and D. Cartwright. 1965. *Structural Models: An Introduction to the Theory of Directed Graphs*. New York: John Wiley.

Heider, F. 1946. ''Attitudes and Cognitive Organization,'' *Journal of Psychology* 21:107-12.

Heil, G. H. and H. C. White. 1976. ''An Algorithm for Finding Simultaneous Homomorphic Correspondences Between Graphs and Their Image Graphs.'' *Behavioral Science* 21:26-35.

Klein, G. and J. E. Aronson. 1991. ''Optimal Clustering: A Model and Method.'' *Naval Research Logistics* 38:447-61.

Koontz, W. L. G., P. M. Narendra, and K. Fukunaga. 1975. ''A Branch and Bound Clustering Algorithm.'' *IEEE Transaction on Computers* C-24:908-15.

Leik, R. K. and B. F. Meeker. 1975. *Mathematical Sociology*. Englewood Cliffs, NJ: Prentice Hall.

Lemann, T. B. and R. L. Solomon. 1952. ''Group Characteristics as Revealed in Sociometric Patterns and Personality Ratings.'' *Sociometry* 15:7-90.

Lorrain, F. and H. C. White. 1971. ''Structural Equivalence of Individuals in Social Networks.'' *Journal of Mathematical Sociology* 1:49-80.

McKinney, J. C. 1948. ''An Educational Application of a Two-Dimensional Sociometric Test.'' *Sociometry* 11:356-67.
Newcomb, T. N. 1961. *The Acquaintance Process*. New York: Holt Rinehart and Winston.
Sailer, L. D. 1978. ''Structural Equivalence: Meaning and Definition, Computation and Application.'' *Social Networks* 1:73-90.
Sampson, S. F. 1968. ''A Novitiate in a Period of Change: An Experimental Case Study of Relationships.'' Ph.D. dissertation, Department of Sociology, Cornell University, Ithaca, NY.
White, D. R. and K. P. Reitz. 1983. ''Graph and Semigroup Homomorphisms on Networks of Relations.'' *Social Networks* 5:193-234.
White, H. C., S. A. Boorman, and R. L. Breiger. 1976. ''Social Structure From Multiple Networks: I. Blockmodels of Roles and Positions.'' *American Journal of Sociology* 81:730-79.

Bios

Michael Brusco is a professor of operations research in the College of Business at Florida State University. His research focuses on the development of exact and approximate algorithms for problems related to scheduling and computational statistics. In this latter area, his work has recently appeared in the *Journal of Mathematical Psychology, Psychological Methods, Psychometrika,* and *Science Magazine*.

Patrick Doreian is professor emeritus of sociology at the University of Pittsburgh and a research faculty member at the Faculty of Social Sciences, University of Ljubljana, Slovenia. He coedits *Social Networks.* Some recent articles of his have appeared in *Social Science and Medicine,* the *Journal of Mathematical Sociology,* and *Social Networks*. He is a coauthor of *Generalized Blockmodeling,* which received the Harrison White Outstanding Book Award in 2007. His primary research interests are in social network analysis.

Andrej Mrvar is an associate professor of social science informatics at the Faculty of Social Sciences, University of Ljubljana, Slovenia. He won several awards for graph drawings at international competitions between 1995 and 2000. Since 2000 he has edited the statistical journal *Metodoloki zvezki*. He is a coauthor (with Vladimir Batagelj) of a widely used program, Pajek, and one of the coauthors of the monograph *Exploratory Social Network Analysis With Pajek* (2005).

Douglas Steinley is an associate professor of psychological sciences at the University of Missouri. His research program concentrates on the development and refinement of modern cluster analytic procedures, including variable selection, variable weighting, and data reduction. In conjunction, he works on partitioning problems from a social networks analysis perspective.