doi:10.1017/nws.2013.5

A variable neighborhood search method for a two-mode blockmodeling problem in social network analysis

MICHAEL BRUSCO

College of Business, Florida State University, Tallahassee, FL, USA
(e-mail: mbrusco@fsu.edu*)*

PATRICK DOREIAN

Department of Sociology, University of Pittsburgh, Pittsburgh, PA, USA
Faculty of Social Sciences, University of Ljubljana, Ljubljana, Slovenia

PAULETTE LLOYD*

AAAS Fellow and US Department of State, Washington, DC, USA

DOUGLAS STEINLEY

Department of Psychological Sciences, University of Missouri–Columbia, Columbia, MO, USA

Abstract

This paper presents a variable neighborhood search (VNS) algorithm that is specially designed for the blockmodeling of two-mode binary network matrices in accordance with structural equivalence. Computational results for 768 synthetic test networks revealed that the VNS heuristic outperformed a relocation heuristic (RH) and a tabu search (TS) method for the same problem. Next, the three heuristics were applied to two-mode network data pertaining to the votes of member countries on resolutions in the United Nations General Assembly. A comparative analysis revealed that the VNS heuristic often provided slightly better criterion function values than RH and TS, and that these small differences in criterion function values could sometimes be associated with substantial differences in the actual partitions obtained. Overall, the results suggest that the VNS heuristic is a promising approach for blockmodeling of two-mode binary networks. Recommendations for extensions to stochastic blockmodeling applications are provided.

Keywords: *deterministic blockmodeling, two-mode binary networks, heuristics, variable neighborhood search, UNGA voting data*

1 Introduction

There are a variety of social network applications characterized by two disjoint sets of objects with index sets, $U = 1, 2, \ldots, N_1$ and $V = 1, 2, \ldots, N_2$, and a proximity function that defines an $N_1 \times N_2$ binary network matrix, $\mathbf{A}$, with elements $a_{ij} = 1$ if there is a tie between objects i and j and $a_{ij} = 0$ otherwise, for all $i \in U$ and $j \in V$.

* Paulette Lloyd is a AAAS fellow and Foreign Affairs Officer at the U.S. Department of State. The views expressed in this paper are her own and do not necessarily reflect those of the Department of State or the United States Government.

Although there are several possible approaches for modeling two-mode network data (Latapy et al., 2008; Wasserman & Faust, 1994), the partitioning of the two object sets is an important method for uncovering structure in such data. Although these two-mode partitioning problems are sometimes identified using different terminology (e.g., biclustering, co-clustering, and generalized blockmodeling), the fact remains that they are relevant to a number of scientific fields including medicine (van Mechelen et al., 2004), biological science (Madeira & Oliveira, 2004; Prelić et al., 2006; van Uitert et al., 2008), psychology (Schepers & van Mechelen, 2011), political science (Doreian et al., 2013; Mische & Pattison, 2000), computer science (Protti et al., 2009), industrial engineering (Selim et al., 1998), and organizational science (Davis et al., 1941; Galaskiewicz, 1985), as well as the general literature on classification (Mirkin et al., 1995; van Rosmalen et al., 2009; Wilderjans et al., 2013).

In this paper, we approach two-mode partitioning from the standpoint of exploratory blockmodeling, which is a procedure that has recently received considerable attention in the social sciences literature (Brusco & Steinley, 2007, 2009, 2011; Brusco et al., 2013; Doreian et al., 2004, 2005). When implemented within the framework of structural equivalence (Lorrain & White, 1971), the goal is to find partitions of both U and V into prespecified numbers of clusters such that the blocks formed by the clusters of the selected partitions are either complete (all 1s) or null (all 0s) to the greatest extent possible. This two-mode blockmodeling formulation results in an inherently challenging discrete optimization problem with a finite but often enormous solution space. Moreover, the problem is closely related to bicluster graph editing, which is known to be NP-hard (Protti et al., 2009).

Although exact solution procedures have been designed for two-mode blockmodeling problems (Brusco et al., 2013; Brusco & Steinley, 2009), these methods are restricted to problems of size 25 × 25 and smaller.[1] In contrast, an approximate procedure designed by Doreian et al. (2004, 2005) generally obtains globally optimal blockmodels for small- to modestly sized networks and is scalable for larger networks. Their approximate procedure is a relocation heuristic (RH) that refines an initial blockmodel by transferring objects from one cluster to another and exchanging the cluster memberships of objects in different clusters. Upon termination, this heuristic yields a blockmodel that is locally optimal with respect to all possible transfers and exchanges, but is not necessarily globally optimal. To increase the chances of finding the global optimum (or, at least, avoiding a poor local optimum), a common strategy is to run multiple repetitions of the RH using different initial blockmodeling solutions.

More recently, Brusco & Steinley (2011) proposed a tabu search (TS) heuristic for deterministic two-mode blockmodeling that substantially outperformed the RH, with respect to minimization of the number of inconsistencies, across 48 large networks. Their results suggest that metaheuristics might have an important role in obtaining high-quality solutions for large two-mode networks in a reasonable amount of time. Our paper further explores this possibility by developing and testing an adaptation of the variable neighborhood search (VNS; Mladenović &

[1] Although networks of this size would be considered very small in some scientific domains, there are many networks in the social sciences literature that fall within these limits (e.g., Davis et al., 1941; Doreian et al., 2004; Galaskiewicz, 1985; Mische & Pattison, 2000).

Hansen, 1997) algorithm for deterministic two-mode blockmodeling. In part, this exploration is motivated by the fact that there is compelling evidence that VNS often outperforms TS in a number of contexts that require the clustering of objects, including the *p*-median problem (Hansen & Mladenović, 1997), minimum sum of squares partitioning (Brusco & Steinley, 2007), and clique partitioning (Brusco & Köhn, 2009).

The efficacy of the VNS algorithm is evaluated via an application to voting data from the United Nations General Assembly (UNGA). In this context, the index set U consists of UNGA member countries, whereas index set V contains UNGA resolutions on which the countries voted. Historically, clique partitioning (Brusco & Köhn, 2009; Grötschel & Wakabayashi, 1989) has been a popular classification approach to UNGA voting data.[2] Conceptually formalized by Règnier (1965), clique partitioning is based on the principle of aggregation of binary equivalence relations. In the context of UNGA voting data, each resolution is used to establish a binary equivalence relation that can be used to separate the countries into maximally homogeneous groups. Thus, together, the resolutions establish a set of partitions for the countries. Règnier's (1965) approach offers an elegant integer linear programming formulation for finding a "median" equivalence relation, conceptualized as a unique partition of the countries, such that the sum of the least squares distances between the median relation and each observed equivalence relation for the resolutions is minimized. Grötschel and Wakabayashi (1989, 1990) provided a clever polyhedral-based reformulation of Règnier's (1965) problem that seeks to partition the vertices (countries) of a graph so as to minimize the sum of the edge weights for the complete subgraphs (or cliques) formed by the partition. They developed a cutting plane procedure for solving the resulting clique partitioning problem, which they applied to a number of UNGA voting data sets.

The obvious limitation of using clique partitioning to cluster UNGA voting data is that only one mode of the data (the countries) is partitioned. The information contained within the resolutions is collapsed to establish a similarity relation among the countries. We propose that two-mode blockmodeling is a preferred approach because it simultaneously establishes partitions for both the countries and the resolutions. Nevertheless, for two-mode blockmodeling to be effective for UNGA voting analyses as well as other large-scale network applications (e.g., those pertaining to supply chains, cognitive diagnosis, social media, purchase behavior, etc.), it is necessary to employ procedures that are capable of obtaining good solutions within a reasonable amount of computation time. Our results indicate that the proposed VNS algorithm is such a method.

In Section 2, we present a formulation for two-mode blockmodeling based on structural equivalence and describe previously published exact and approximate procedures for obtaining solutions. Section 3 presents a new approach to the two-mode blockmodeling problem based on the VNS algorithm. A computational evaluation of the proposed algorithm via comparison to the relocation and TS heuristics is presented in Section 4. An illustrative example of the VNS algorithm

[2] Using spatial models is a popular approach in political science; see Clinton et al. (2004); Poole & Rosenthal (1985); Poole (2005); Poole et al. (2011). Factor analysis is another popular approach; see Kim & Russett (1996) and Voeten (2000). For a blockmodeling approach, see Doreian et al. (2013).

within the context of United Nations voting data is provided in Section 5. The paper concludes in Section 6 with a summary of findings and identification of extensions for future research.

2 Two-mode blockmodeling based on structural equivalence

2.1 Formulation of two-mode blockmodeling

To facilitate our description of two-mode blockmodeling, we recall that **A** is an $N_1 \times N_2$ binary data matrix with elements $a(i,j) = 1$ if there is a tie between row object i and column object j and $a(i,j) = 0$ otherwise, for $i \in U$ and $j \in V$. The desired numbers of clusters for the row objects and column objects are k_1 and k_2, respectively. We define Π as the set of all k_1-cluster partitions of $U, \pi = R_1, R_2, \ldots, R_{k_1}$ as a specific k_1-cluster partition in Π where R_k contains the indices of the row objects assigned to cluster k for $1 \leqslant k \leqslant k_1$, and $|R_k|$ as the cardinality of R_k for $1 \leqslant k \leqslant k_1$. Similarly, Ω is the set of all k_2-cluster partitions of V, $\omega = C_1, C_2, \ldots, C_{k_2}$ is a specific k_2-cluster partition in Ω where C_l contains the indices of the column objects assigned to cluster l for $1 \leqslant l \leqslant k_2$, and $|C_l|$ is the cardinality of C_l for $1 \leqslant l \leqslant k_2$. With these definitions in place, the formulation of the discrete optimization problem for two-mode blockmodeling can be written as follows:

$$\underset{(\pi \in \Pi, \omega \in \Omega)}{\text{Minimize}} : g(\pi, \omega) = \sum_{k=1}^{k_1} \sum_{l=1}^{k_2} \min\{\lambda_{kl}, \rho_{kl}\}, \tag{1}$$

where

$$\lambda_{kl} = \sum_{i \in R_k} \sum_{j \in C_l} a(i,j), \qquad \forall\, 1 \leqslant k \leqslant k_1 \text{ and } 1 \leqslant l \leqslant k_2, \tag{2}$$

and

$$\rho_{kl} = \sum_{i \in R_k} \sum_{j \in C_l} (1 - a(i,j)), \qquad \forall\, 1 \leqslant k \leqslant k_1 \text{ and } 1 \leqslant l \leqslant k_2. \tag{3}$$

The objective function (1) of this optimization problem is the total number of inconsistencies in contrast to an ideal structure where all blocks are either complete or null for which there are no inconsistencies. The problem requires the identification of a k_1-cluster partition of the row objects, π, and a k_2-cluster partition of the column objects, ω, that minimizes this function. The values of λ_{kl} and ρ_{kl} represent the number of 1s and 0s, respectively in the block defined by the objects in row cluster k and column cluster l. If $\lambda_{kl} \geqslant \rho_{kl}$, then the block is complete and ρ_{kl} is the number of inconsistencies in the block; otherwise, if $\lambda_{kl} < \rho_{kl}$, then the block is null and λ_{kl} is the number of inconsistencies in the block.

2.2 Existing methods

An exact solution procedure for the two-mode blockmodeling problem in Section 2.1 was developed by Brusco et al. (2013). The algorithm is a depth-first search branch-and-bound algorithm that enters the row and column objects in alternating fashion during the search process. The algorithm is computationally feasible for problems where the total number of row and column objects is no more than 50.

Doreian et al. (2004) developed a RH for two-mode blockmodeling that uses two local search operations. The first operation examines all possible *transfers* of an object from its current cluster to one of the other clusters. The second operation considers all possible *exchanges* of two objects not currently in the same cluster. In two-mode blockmodeling, transfers and exchanges are evaluated for both the row objects and the column objects. Relocation algorithms can be implemented in either an "on-the-fly" or "greedy" fashion. In the on-the-fly implementation, each partition that improves the objective function during the search process is immediately accepted as it is found. In the greedy implementation, all transfers and exchanges are evaluated without accepting any new partition until the transfer or exchange that yields the greatest improvement in the objective function establishes a partition that is selected as the new partition. On-the-fly implementations are typically faster, whereas greedy implementations often avoid accepting a less profitable move when a better one is available. Both the on-the-fly and greedy implementations terminate when no transfers or exchanges further improve the objective function. To avoid the potential for a poor local optimum, relocation algorithms should be restarted multiple times from different initial partitions.

Brusco & Steinley (2011) devised a TS heuristic for the two-mode blockmodeling problem. This algorithm embeds the RH within the TS framework, allowing for escape from local optima by accepting neighborhood search moves that worsen the objective function. Brusco and Steinley compared two different multiple restart versions of their TS heuristic to two different multiple restart versions of the RH. For both algorithms, the versions were differentiated by the local search operations used: (a) one version used both transfers and exchanges, and (b) one version used transfers only. For both the TS and the RH heuristics, the versions using exchanges were too computationally costly and did not foster the discovery of better solutions. Both algorithms performed better using only transfers because this enabled them to achieve many more restarts within the fixed 10-minute time limit for the algorithms. Overall, the multiple restart TS procedure using only transfers substantially outperformed all other competing methods across a set of large, randomly generated matrices up to size 400 × 400.

3 The variable neighborhood search algorithm

VNS is a solution procedure proposed by Mladenović & Hansen (1997) that relies on the systematic variation of the neighborhood within the framework of the local search process. The procedure has been successfully implemented in a wide variety of combinatorial data analysis applications. Most relevant among these for our purposes are applications to other clustering problems. For example, Hansen & Mladenović (1997) applied VNS to p-median clustering, and Hansen & Mladenović (2001) adapted the method for K-means clustering. Brusco & Köhn (2009) devised a procedure similar to VNS for the clique partitioning problem.

The pseudo code for the main program of the VNS algorithm we developed for the problem posed in Section 2.1 is provided in Figure 1. Embedded within the VNS framework are two major subroutines: (1) INITIALIZE, which is a procedure that establishes an initial blockmodel and (2) RELOCATION, which is the RH

```
Call the INITIALIZE subroutine to obtain $\pi^*$ and $\omega^*$.
Set $ymin = 0.05$, $ymax = 0.5$, $ystep = 0.01$, $ypert = ymin$, $\pi = \pi^*$, $\omega = \omega^*$
While $ypert \le ymax$
        Call the RELOCATION subroutine to refine $\pi$ and $\omega$.
        % Adjust the Perturbation Parameter
        if $g(\pi,\omega) < g(\pi^*,\omega^*)$
                $g(\pi^*,\omega^*) = g(\pi,\omega)$
                $\pi^* = \pi$
                $\omega^* = \omega$
                $ypert = ymin$
        else
                $ypert = ypert + ystep$
        end if
        % Check the Termination Condition
        if $ypert > ymax$
                continue (i.e., pass control to End While below)
        end
        % Perturb the Incumbent Solution
        $\pi = \pi^*$
        $\omega = \omega^*$
        For $i = 1$ to $N_1$
                $rnd$ = uniform random number on the interval [0, 1]
                if $rnd < ypert$
                        $k' = k : i \in R_k$
                        $k''$ a cluster randomly selected from the interval $(1 \le k \le k_1) : k'' \ne k'$
                        $R_{k'} = R_{k'} \setminus \{i\}$
                        $R_{k''} = R_{k''} \cup \{i\}$
                end if
        Next $i$
        For $j = 1$ to $N_2$
                $rnd$ = uniform random number on the interval [0,1]
                if $rnd < ypert$
                        $l' = l : j \in C_l$
                        $l''$ a cluster randomly selected from the interval $(1 \le l \le k_2) : l'' \ne l'$
                        $C_{l'} = C_{l'} \setminus \{j\}$
                        $C_{l''} = C_{l''} \cup \{j\}$
                end if
        Next $j$
End While
Return $\pi^*$ and $\omega^*$
```

Fig. 1. The pseudo code for the main VNS program.

that is used to refine solutions. We provide a description of these subroutines in the following subsections.

3.1 The INITIALIZE subroutine

Some implementations (e.g., Batagelj & Mrvar, 1998; Brusco & Steinley, 2007; Doreian et al., 2004) of blockmodeling algorithms use purely random assignment of objects to clusters to establish initial partitions. The fundamental problem with purely random assignment as an initialization procedure is that it has no proclivity for getting the relocation process off to a good start. Instead, the analyst is left to hope that, by using a large number of repetitions, some of those repetitions will provide initial solutions that are conducive to finding the global optimum. For some networks, particularly very sparse (or very dense) networks, this might not occur even with numerous repetitions. To remedy this problem, we propose an initialization

```
Set $R_k = \varnothing \ \forall \ 1 \le k \le k_1$, $C_l = \varnothing \ \forall \ 1 \le l \le k_2$
Set $d(k, j) = 0 \ \forall \ 1 \le k \le k_1$ and $\forall \ 1 \le j \le N_2$, and set $e(i, l) = 0 \ \forall \ 1 \le i \le N_1$ and $\forall \ 1 \le l \le k_2$
% Randomly Select $k_1$ Row Exemplars and Assign Each Row Object to its Nearest Exemplar
For $k = 1$ to $k_1$
    randomly select a row object, $h : \{a(h, j) \neq d(q, j) \ \forall \ 1 \le j \le N_2\}$ for any $q$ $(1 \le q \le k\text{-}1)$.
    For $j = 1$ to $N_2$
        $d(k, j) = a(h, j)$
    Next $j$
Next $k$
For $i = 1$ to $N_1$
    $mindist = \infty$
    For $k = 1$ to $k_1$
        $dist = 0$
        For $j = 1$ to $N_2$
            $dist = dist + |a(i, j) - d(k, j)|$
        Next $j$
        if $dist < distmin$
            $distmin = dist$
            $k' = k$
        end if
    next $k$
    $R_k = R_{k'} \cup \{i\}$
next $i$
% Randomly Select $k_2$ Column Exemplars and Assign Each Column Object to its Nearest Exemplar
For $l = 1$ to $k_2$
    randomly select a column object, $p : \{a(i, p) \neq e(i, r) \ \forall \ 1 \le i \le N_1\}$ for any $r$ $(1 \le r \le l\text{-}1)$.
    For $i = 1$ to $N_1$
        $e(i, l) = a(i, p)$
    Next $i$
Next $l$
For $j = 1$ to $N_2$
    $mindist = \infty$
    For $l = 1$ to $k_2$
        $dist = 0$
        for $i = 1$ to $N_1$
            $dist = dist + |a(i, j) - e(i, l)|$
        next $i$
        if $dist < distmin$
            $distmin = dist$
            $l' = l$
        end if
    next $l$
    $C_l = C_{l'} \cup \{j\}$
next $i$
Set $\pi^* = \{R_1, \ldots R_{k1}\}$
Set $\omega^* = (C_1, \ldots, C_{k2}\}$
Return $\pi^*$ and $\omega^*$
```

Fig. 2. The pseudo code for the INITIALIZE subroutine.

procedure incorporating some degree of randomness but also containing a logical component corresponding to the construction of the initial blockmodel. The pseudo code for subroutine INITIALIZE, which implements the initialization procedure, is provided in Figure 2.

The procedure begins with the random selection of k_1 and k_2 "exemplars" for row and column objects, respectively. Care is taken in the random selection process to assure that none of the selected exemplars are identical. For example, no two row exemplars should have exactly the same column elements across all columns. The exemplars serve as provisional prototypes for their cluster. In the next step, each row object is assigned to the cluster of the row exemplar to which it is nearest based

```
Input $\pi$ and $\omega$ and set $flag = 1$.
While $flag = 1$
    $flag = 0$
    % Check All Transfers of Row Objects to Different Clusters
    For $i = 1$ to $N_1$
        $h = h : i \in R_h$
        if $|R_h| > 1$
            for $k = 1$ to $k_1$
                if $k \neq h$ then
                    $\pi' = \{\pi \setminus \{R_k, R_h\}\} \cup \{R_k \cup \{i\}\} \cup \{R_h \setminus \{i\}\}$
                    if $f(\pi', \omega) < f(\pi, \omega)$
                        $\pi = \pi'$
                        $flag = 1$
                    end if
                end if
            next $k$
        end if
    next $i$
    % Check All Transfers of Column Objects to Different Clusters
    For $j = 1$ to $N_2$
        $p = p : j \in C_p$
        if $|C_p| > 1$
            for $l = 1$ to $k_2$
                if $l \neq p$ then
                    $\omega' = \{\omega \setminus \{C_l, C_p\}\} \cup \{C_l \cup \{j\}\} \cup \{C_p \setminus \{j\}\}$
                    if $f(\pi, \omega') < f(\pi, \omega)$
                        $\omega = \omega'$
                        $flag = 1$
                    end if
                end if
            next $k$
        end if
    next i
End While
Return $\pi$ and $\omega$
```

Fig. 3. The pseudo code for the RELOCATION subroutine.

on an absolute distance measure (any ties are broken based on the first exemplar that obtains the minimum distance). Similarly, each column object is assigned to the cluster of the column exemplar to which it is nearest. Upon termination, the initialization procedure passes the row partition (π^*) and column partition (ω^*) back to the main VNS program.

3.2 The RELOCATION subroutine

Our adaptation of the VNS algorithm uses an on-the-fly implementation of the Doreian et al. (2004) RH with transfers only. The pseudo code for the RELOCATION subroutine is provided in Figure 3.

The RELOCATION subroutine requires row and column partitions (π and ω) as input. A *flag* variable that controls termination of the algorithm is initially set to one. Immediately after entering the "While" loop, we set *flag* = 0. Next, all transfers of row objects are tested. For each object $i \in U$, the effect on the objective criterion function for the transfer of that object from its current cluster, h, to one of the other clusters, $1 \leqslant k \neq h \leqslant k_1$ is evaluated. Each time a transfer improves the objective function, it is immediately accepted and the *flag* is set to one to indicate that an improvement in the objective function was found. This process of transfer evaluation is then repeated for the column objects. If *flag* $= 1$ at the "End While" statement, then there was at least one row-object or column-object transfer that produced an improvement in the objective criterion function on the last cycle. In such cases, another pass through the "While" loop is made. Contrastingly, if *flag* $=$ 0 at the "End While" statement, then no improvement was realized on the last cycle and the algorithm terminates.

The relocation algorithm is "left-biased" in the sense that objects earlier (i.e., to the left) in the list have an opportunity to be relocated first. The left bias pertains not only within row and column objects, but also with respect to the fact that the row object relocations are tested first.

3.3 The main VNS program

The main VNS algorithm begins with a call to the INITIALIZE subroutine to provide initial best-found partitions, π^* and ω^*, which are then specified as the current (incumbent) partitions (i.e., $\pi = \pi^*$ and $\omega = \omega^*$). The size of the neighborhood explored is controlled by *ypert*, which is adapted during the algorithm via three parameters: *ymin*, *ymax*, and *ystep*, which are subsequently initialized. Together, these parameters control the probability of relocating objects across clusters when perturbing the best-found solution. As the names suggest, *ymin* is the minimum probability, *ymax* is the maximum probability, and *ystep* is the size of the progression steps from *ymin* to *ymax*. Our experimentation has shown that the performance of the VNS algorithms is fairly robust with respect to parameter selection; however, a reasonable choice for these parameters, which is used throughout this paper, is *ymin* $= 0.05$, *ymax* $= 0.5$, and *ystep* $= 0.01$.

Initially, *ypert* is set to *ymin*. The algorithm will continue the VNS until *ypert* exceeds *ymax*, as shown in the main "While" loop in Figure 1. Upon entering the "While" loop, the RELOCATION subroutine is called to refine the current incumbent solution. If this refinement process produces a new best-found solution, then that solution is installed as the new best-found solution and *ypert* is set to *ymin*; otherwise, *ypert* is incremented by *ystep*. As noted previously, the algorithm terminates when *ypert* exceeds *ymax*. If *ypert* $<$ *ymax*, then the best-found solution is re-installed as the incumbent solution, and that incumbent solution is subsequently perturbed by randomly adjusting the cluster memberships of row and column objects. During this perturbation phase, the probability of adjusting the cluster membership of each row and column object is *ypert*. For each object, a uniform [0,1] random number is drawn, and if the random number is less than *ypert*, then the object is relocated to a new randomly selected cluster. After perturbation, control returns to the top of the "While" loop and the RELOCATION subroutine is applied to refine the newly perturbed solution.

There are a couple of salient observations that should be made regarding the VNS heuristic. First, if *ymax* and *ymin* are set to zero, then the VNS algorithm reduces to the RH with transfers only, whereby no perturbation occurs and the heuristic terminates when a locally optimal solution (with respect to all possible transfers) is obtained. Second, like the relocation and TS heuristics evaluated by Brusco &

Steinley (2011), the VNS is sensitive to the quality of the initial starting solution and, accordingly, is well suited to a multiple restart implementation.

4 Computational study

4.1 Test problems

A computational evaluation of the VNS algorithm was performed using a suite of 768 test problems from a study by Brusco & Steinley (2007). The test problems were constructed by modifying each of eight design features at two different levels. The first feature, the number of row objects, was tested at levels of $N_1 = 60$ and $N_1 = 120$. The second feature, the number of column objects, was tested at levels of $N_2 = 60$ and $N_2 = 120$. The third and fourth design features were the number of row clusters (tested at levels of $k_1 = 3$ and $k_1 = 6$) and the number of column clusters (tested at levels of $k_2 = 3$ and $k_2 = 6$), respectively. The fifth design feature was the distribution of the row objects across the row clusters. At one level of the fifth design feature, the number of objects in each cluster was equal. For the second level of the fifth design feature, the first cluster contained 60% of the objects and the remaining objects were equally spread among the remaining $k_1 - 1$ clusters. The sixth design feature was the distribution of the column objects across the column clusters, with levels concordant with those of the fifth design feature. The seventh design feature was the probability for "near-complete" blocks,[3] which was tested at levels of 50% and 70%. For example, at the second level of this design feature, the probability of each block being assigned "near-complete" status was 70%, and thus there was a 30% chance of the block being "near-null." The eighth design feature pertained to "block strength," which is designed as the probability for a proper matrix element in the block. At the first level of this design feature, for each near-complete (near-null) block, the probability for each element in the block assuming a value of 1 (0) was 90%. The block strength for the second level was reduced by using a probability of 75%. Three test problem replicates were generated for each of the $2^8 = 256$ cells of the design, resulting in a set of 768 unique test problems.

Figure 4 is helpful for illustrating the network generation process. Consider parameter settings of $N_1 = N_2 = 60, k_1 = k_2 = 3$, level 2 for the distribution of row objects, level one (even) distribution for the column objects, 70% probability for "near-complete" blocks, and 90% block strength. As shown in Figure 4, there are a total of nine blocks formed, and they are of different sizes because the cluster sizes for the row objects are not equal. The selection of which blocks are near-complete and which blocks are near-null is random, but biased in favor of near-complete (70% chance). Within each near-complete (near-null) block, the elements are randomly generated with a 90% probability of assuming a value of one (zero). After generating the network matrix, the rows and columns are randomly perturbed.

[3] We use the terms "near-complete" and "near-null" to indicate that the generated blocks are well structured (with high percentages of 1s or 0s as appropriate), but not perfectly complete (all 1s) or perfectly null (all 0s).

Column Objects

Row objects	1, 2, … …17, 18, 19, 20	21, 22, 23 …. …37, 38, 39, 40	41, 42,. 43, 44 …57, 58, 59, 60
1 2 36	**Near-Complete** Each element in this block has a 90% probability of assuming a value of 1, and a 10% chance of assuming a value of 0.	**Near-Null** Each element in this block has a 10% probability of assuming a value of 1, and a 90% chance of assuming a value of 0.	**Near-Complete** Each element in this block has a 90% probability of assuming a value of 1, and a 10% chance of assuming a value of 0.
37 38 … … 48	**Near-Complete** Each element in this block has a 90% probability of assuming a value of 1, and a 10% chance of assuming a value of 0.	**Near-Complete** Each element in this block has a 90% probability of assuming a value of 1, and a 10% chance of assuming a value of 0.	**Near-Complete** Each element in this block has a 90% probability of assuming a value of 1, and a 10% chance of assuming a value of 0.
49 50 … … 60	**Near-Complete** Each element in this block has a 90% probability of assuming a value of 1, and a 10% chance of assuming a value of 0.	**Near-Null** Each element in this block has a 10% probability of assuming a value of 1, and a 90% chance of assuming a value of 0.	**Near-Null** Each element in this block has a 10% probability of assuming a value of 1, and a 90% chance of assuming a value of 0.

Fig. 4. An example of the data generation structure for the simulation experiment: $N_1 = N_2 = 60$, $k_1 = k_2 = 3$, 60% of the row objects in the first row cluster (equal spread of row objects among the remaining row clusters), an even distribution of column objects across the column clusters, 70% probability for near-complete blocks, and 90% block strength. The near-complete and near-null blocks were selected at random using a 70% probability for near-complete and 30% for near-null.

4.2 Algorithm implementation

The VNS algorithm is written in Fortran 90 and was implemented on a microcomputer with a 2.2 GHz Pentium 4 PC with 1 GB of RAM. The VNS algorithm was applied to each of the 768 test problems using the parameter settings noted above ($ymin = 0.05$, $ymax = 0.50$; $ystep = 0.01$), with a time limit of 20 CPU seconds imposed. For example, the first run was for test problem 1, using parameter settings, and the algorithm was permitted to restart as many times as necessary provided that no more than 20 seconds have elapsed. The best performing RH and TS algorithm versions from Brusco & Steinley's (2011) study were implemented with a 20-second time limit for comparative purposes. The best criterion function value obtained by each of the three methods (RH, TS, and VNS) within the 20-second time limit was stored for each algorithm. In light of the fact that global optima are not available, the best criterion function value across the three methods (denoted as

Table 1. *Computational results for the simulation study.*

	Mean percentage deviation above the best criterion value across all three methods			Mean percentage of test problems for which the best criterion function across all three methods was obtained		
Design feature settings	RH	TS	VNS	RH	TS	VNS
60 row objects	0.142	0.310	0.007	75.5	72.4	98.4
120 row objects	0.132	0.736	0.034	78.9	67.2	97.9
60 column objects	0.123	0.423	0.014	76.0	70.3	98.4
120 column objects	0.151	0.623	0.027	78.4	69.3	97.9
3 row clusters	0.026	0.067	0.006	86.5	82.0	99.2
6 row clusters	0.248	0.980	0.035	68.0	57.6	97.1
3 column clusters	0.024	0.071	0.003	89.1	82.8	98.7
6 column clusters	0.251	0.976	0.038	65.4	56.8	97.7
Even row object distribution	0.088	0.283	0.005	82.0	77.3	98.7
60% row object distribution	0.187	0.763	0.036	72.4	62.2	97.7
Even column object distribution	0.076	0.370	0.008	80.7	74.2	98.7
60% column object distribution	0.198	0.676	0.033	73.7	65.4	97.7
50% complete blocks	0.109	0.509	0.022	75.5	65.9	97.7
70% complete blocks	0.165	0.538	0.019	78.9	73.7	98.7
90% block strength	0.082	0.676	0.021	92.4	83.9	99.2
75% block strength	0.193	0.371	0.020	62.0	55.7	97.1
Overall measures	0.137	0.520	0.020	77.2	69.8	98.2

RH = relocation heuristic; TS = tabu search; VNS = variable neighborhood search.

g^*) represented the benchmark for each test problem. Accordingly, the performance of each heuristic was measured as the percentage deviation above this benchmark for each test problem.

4.3 Experimental results

The results of the experimental study are reported in Table 1. For each of the three heuristic methods (RH, TS, and VNS), and each level of each design feature, Table 1 displays the mean percentage deviation above the benchmark criterion value (g^*) and the mean percentage of test problems for which the benchmark criterion value was obtained. The VNS algorithm matched the benchmark criterion value for 98.2% (754 out of 768) of the test problems. This was substantially better than its two competitors, RH and VNS, which matched the benchmark criterion value for only 77.2% (593 out of 768) and 69.8% (536 out of 768) of the test problems, respectively. The VNS algorithm was also appreciably better than RH and TS with respect to the mean percentage deviation above the benchmark solutions. The overall mean percentage deviations for RH, TS, and VNS were 0.137%, 0.520%, and 0.020%, respectively. An analysis of variance was conducted using percentage deviation above the benchmark criterion function value as the dependent variable. The eight design features served as blocking factors, whereas the solution method was the factor of interest. The null hypothesis of equal means among the three methods was rejected ($p < 0.001$). Moreover, pairwise comparisons of the RH, TS, and VNS means using Tukey's method revealed significant difference among all pairs ($p < 0.05$).[4]

The results for the different levels of the design features reveal that the superiority of the VNS heuristic spanned all problem characteristics. The VNS algorithm obtained the benchmark criterion function value for at least 97.1% of the test problems at each design feature level. Contrastingly, the RH and TS algorithms struggled mightily at some design feature levels. Most notably, "block strength" appeared to have a profound effect on the performance of the RH and TS methods. At the 90% block strength setting, the mean percentages of test problems for which the benchmark criterion function values were obtained were 99.2%, 92.4%, and 83.9% for VNS, RH, and TS, respectively. At the 75% block strength level, the VNS algorithm obtained the benchmark for a healthy 97.1% of the problems, whereas the RH and TS percentages plummeted to 62.0% and 55.7%, respectively. The conclusion from these results is that the VNS heuristic is far more robust with respect to degradation in block structure. The superiority of the VNS heuristic also was more pronounced when the number of row (or column) clusters was increased from three to six, and when the distribution of row (or column) objects across clusters was changed from an even distribution to 60% in one cluster and an equal spread among the remaining clusters.

5 Application to UNGA voting data

5.1 Empirical data sets

We noted at the outset that the kinds of partitioning efforts that formed our point of departure used relatively small data sets. In order to assess the method introduced here more stringently, larger two-mode data sets were necessary. We used data originally collected and coded by Lloyd including partitions created for an earlier blockmodeling study (Doreian et al., 2013). The data included all UNGA roll call votes from 1981 to 2000, which we partitioned into four time periods (5 and 10 years before and after the end of the Cold War) and four issue types (ideological, military, political, and economic, roughly corresponding to the issues discussed in the Main Committees of the UNGA). We selected the military and ideological resolutions for the period 1996 to 2000, from which we created two network data sets of different sizes. The first network corresponds to the votes of $N_1 = 153$ member countries on $N_2 = 129$ military-issue resolutions. The elements of the 153×129 network matrix are $a_{ij} = 1$ if member country i voted in favor of military resolution j and 0 otherwise.[5] The second network is similar to the first, except that the resolutions are ideologically oriented rather than military oriented. There were $N_1 = 141$ member countries and $N_2 = 272$ ideological resolutions associated with the second network matrix.

[4] The analysis of variance was repeated after logarithmic transformation of the dependent variable (i.e., $\ln(y + 1)$), where y is the original dependent variable measure); however, this did not affect the conclusions of the significance test results for the methods. We also ran a one-way Kruskal–Wallis non-parametric comparison of the results for the three methods and found significant differences in their performance.

[5] For the purpose of this study, we collapsed categories of non-support (no, abstain, and absent) into the category "no" indicating not supporting a resolution. Prior studies have indicated that countries choose between accepting or rejecting resolutions rather than evaluate them (Voeten, 2000).

Table 2. *Results for UNGA military resolutions network—criterion function values.*

		$k_2 = 2$	$k_2 = 3$	$k_2 = 4$	$k_2 = 5$	$k_2 = 6$	$k_2 = 7$
$k_1 = 2$	RH	**1,945**	**1,945**	**1,945**	**1,945**	**1,945**	**1,945**
	TS	**1,945**	**1,945**	**1,945**	**1,945**	**1,945**	**1,945**
	VNS	**1,945**	**1,945**	**1,945**	**1,945**	**1,945**	**1,945**
$k_1 = 3$	RH	**1,945**	**1,829**	**1,826**	**1,826**	**1,826**	**1,826**
	TS	**1,945**	**1,829**	**1,826**	**1,826**	**1,826**	**1,826**
	VNS	**1,945**	**1,829**	**1,826**	**1,826**	**1,826**	**1,826**
$k_1 = 4$	RH	**1,945**	**1,805**	**1,743**	**1,730**	**1,730**	**1,730**
	TS	**1,945**	**1,805**	**1,743**	**1,730**	**1,730**	**1,730**
	VNS	**1,945**	**1,805**	**1,743**	**1,730**	**1,730**	**1,730**
$k_1 = 5$	RH	**1,945**	**1,805**	**1,713**	**1,663**	*1,657*	*1,650*
	TS	**1,945**	**1,805**	**1,713**	**1,663**	**1,649**	**1,646**
	VNS	**1,945**	**1,805**	**1,713**	**1,663**	**1,649**	*1,649*
$k_1 = 6$	RH	**1,945**	**1,805**	**1,707**	**1,633**	*1,619*	*1,613*
	TS	**1,945**	**1,805**	*1,709*	**1,633**	**1,612**	*1,608*
	VNS	**1,945**	**1,805**	*1,709*	**1,633**	**1,612**	**1,599**
$k_1 = 7$	RH	**1,945**	**1,805**	**1,707**	*1,634*	*1,588*	*1,566*
	TS	**1,945**	**1,805**	**1,707**	**1,627**	**1,577**	**1,565**
	VNS	**1,945**	**1,805**	**1,707**	*1,630*	**1,577**	**1,565**

Note. Cell entries in bold indicate the best criterion function value found across all methods, whereas values in italics indicate inferior criterion function values. Abbreviations are explained in the note to Table 1.

5.2 Implementation of methods

The VNS algorithm was applied to both of the UNGA networks for all combinations of k_1 and k_2 on the intervals $2 \leqslant k_1 \leqslant 7$ and $2 \leqslant k_2 \leqslant 7$. For each combination of k_1 and k_2, a 10-minute time limit was used. The TS heuristic and the RH were also applied to both networks using a 10-minute limit for each combination of k_1 and k_2. Accordingly, the comparative analysis corresponds to two networks, 36 combinations of k_1 and k_2, and three methods (VNS, TS, and RH). The principal basis of comparison for the three methods is the criterion function value achieved, which is what all three methods seek to optimize. However, using Hubert & Arabie's (1985) adjusted Rand index (ARI), we also measure the agreement of the country and resolution partitions obtained by these methods. An ARI of value of 0 reflects chance agreement, whereas a value of 1 shows perfect agreement. Steinley (2004) offers ARI guidelines whereby 0.65, 0.8, and 0.9 indicate fair, good, and excellent agreement, respectively. As a final evaluation, we compare the results of the VNS algorithm to those obtained using Pajek, which is a popular software package for generalized blockmodeling (see Batagelj & Mrvar, 1998; Batagelj et al., 2004).

5.3 Comparison of RH, TS, and VNS methods

The RH, TS, and VNS criterion function values for the UNGA military resolutions network are reported in Table 2. The criterion function values of the three algorithms for the UNGA ideological resolutions network are provided in Table 3. What is immediately striking about Tables 2 and 3 is that the relative performances of the three algorithms are much more competitive for these two real-world networks.

Table 3. *Results for UNGA ideological resolutions network—criterion function values.*

		$k_2 = 2$	$k_2 = 3$	$k_2 = 4$	$k_2 = 5$	$k_2 = 6$	$k_2 = 7$
$k_1 = 2$	RH	**5,487**	**5,475**	**5,475**	**5,475**	**5,475**	**5,475**
	TS	**5,487**	**5,475**	**5,475**	**5,475**	**5,475**	**5,475**
	VNS	**5,487**	**5,475**	**5,475**	**5,475**	**5,475**	**5,475**
$k_1 = 3$	RH	**5,441**	**4,791**	**4,709**	**4,671**	**4,671**	**4,671**
	TS	**5,441**	**4,791**	**4,709**	**4,671**	*4,675*	*4,674*
	VNS	**5,441**	**4,791**	**4,709**	*4,672*	**4,671**	**4,671**
$k_1 = 4$	RH	**5,441**	**4,765**	**4,220**	**4,144**	**4,136**	**4,136**
	TS	**5,441**	**4,765**	**4,220**	**4,144**	*4,144*	**4,136**
	VNS	**5,441**	**4,765**	**4,220**	**4,144**	**4,136**	**4,136**
$k_1 = 5$	RH	**5,441**	**4,765**	**4,200**	**4,020**	*3,950*	*3,896*
	TS	**5,441**	**4,765**	**4,200**	**4,020**	**3,947**	*3,947*
	VNS	**5,441**	**4,765**	**4,200**	**4,020**	**3,947**	**3,890**
$k_1 = 6$	RH	**5,441**	**4,765**	*4,198*	**4,001**	**3,841**	*3,772*
	TS	**5,441**	**4,765**	**4,194**	**4,001**	**3,841**	**3,763**
	VNS	**5,441**	**4,765**	*4,196*	**4,001**	**3,841**	**3,763**
$k_1 = 7$	RH	**5,441**	**4,765**	**4,194**	*4,001*	**3,822**	*3,695*
	TS	**5,441**	**4,765**	**4,194**	*3,999*	*3,825*	**3,691**
	VNS	**5,441**	**4,765**	*4,196*	**3,998**	**3,822**	**3,691**

Note. Cell entries in bold indicate the best criterion function value found across all methods, whereas values in italics indicate inferior criterion function values. Abbreviations are explained in the note to Table 1.

In contrast to the synthetic networks in Section 4, all three methods yielded the same criterion function value for most of the test problems. Specifically, of the 72 test problems associated with two networks and 36 combinations of k_1 and k_2, the RH, TS, and VNS algorithms obtained the same criterion function in 52 (72.2%) instances.

Denoting g^* as the best-found criterion function value (across all three methods) for a given test problem, RH, TS, and VNS algorithms matched g^* for 59 (81.9%), 64 (88.9%), and 66 (91.7%) test problems, respectively. Accordingly, on this measure, the rank-ordered performance of the three methods is same as it was for the synthetic problems in Section 4; however, the differences are much more subtle.

For those problems where an algorithm failed to obtain g^*, we computed the percentage error associated with its criterion function. This enabled an average percentage error to be computed for each algorithm. The average percentage errors for the RH, TS, and VNS algorithms were 0.055%, 0.036%, and 0.008%, respectively. Moreover, the VNS algorithm had a maximum percentage of only 0.184%, a figure that was exceeded seven times by RH (maximum of 0.875%) and three times by TS (maximum of 1.465%). These results suggest that the VNS algorithm was much less vulnerable to substantial departures from the best-found criterion function values.

Tables 4 and 5 report the ARI values among the partitions of the three algorithms for countries and military resolutions, respectively. Similarly, Tables 6 and 7 provide the ARI values among the partitions of the three algorithms for countries and ideological resolutions, respectively. An immediately noticeable aspect of Tables 4–7 is the substantial disparity in ARI values across the combinations of k_1 and k_2. Several aspects are particularly important:

Table 4. *Results for UNGA military resolutions network—ARI values for countries.*

		$k_2 = 2$	$k_2 = 3$	$k_2 = 4$	$k_2 = 5$	$k_2 = 6$	$k_2 = 7$
$k_1 = 2$	RH–TS	1.000	1.000	1.000	1.000	1.000	1.000
	RH–VNS	1.000	1.000	1.000	1.000	1.000	1.000
	TS–VNS	1.000	1.000	1.000	1.000	1.000	1.000
$k_1 = 3$	RH–TS	1.000	1.000	1.000	1.000	1.000	1.000
	RH–VNS	1.000	1.000	1.000	1.000	1.000	1.000
	TS–VNS	1.000	1.000	1.000	1.000	1.000	1.000
$k_1 = 4$	RH–TS	0.983	1.000	1.000	1.000	1.000	1.000
	RH–VNS	0.465	1.000	0.966	1.000	1.000	1.000
	TS–VNS	0.470	1.000	0.966	1.000	1.000	1.000
$k_1 = 5$	RH–TS	1.000	1.000	1.000	0.931	0.954	0.930
	RH–VNS	0.383	0.797	1.000	0.948	0.954	0.984
	TS–VNS	0.383	0.797	1.000	0.911	1.000	0.945
$k_1 = 6$	RH–TS	1.000	0.469	0.802	1.000	0.740	0.729
	RH–VNS	0.328	0.594	0.979	1.000	0.743	0.902
	TS–VNS	0.328	0.624	0.820	1.000	0.985	0.782
$k_1 = 7$	RH–TS	1.000	1.000	0.789	0.859	0.861	0.969
	RH–VNS	0.291	0.510	0.988	0.850	0.876	0.715
	TS–VNS	0.291	0.510	0.791	0.990	0.985	0.709

Note. The convention METHOD1–METHOD2 refers to the agreement between the METHOD1 and METHOD2 partitions. For example, RH–TS refers to the agreement of partitions obtained by the RH and TS methods. Abbreviations are explained in the note to Table 1.

Table 5. *Results for UNGA military resolutions network—ARI values for military resolutions.*

		$k_2 = 2$	$k_2 = 3$	$k_2 = 4$	$k_2 = 5$	$k_2 = 6$	$k_2 = 7$
$k_1 = 2$	RH–TS	1.000	1.000	1.000	1.000	1.000	1.000
	RH–VNS	1.000	1.000	1.000	1.000	1.000	1.000
	TS–VNS	1.000	1.000	1.000	1.000	1.000	1.000
$k_1 = 3$	RH–TS	1.000	1.000	1.000	0.659	1.000	0.661
	RH–VNS	1.000	1.000	1.000	0.663	0.828	0.589
	TS–VNS	1.000	1.000	1.000	0.652	0.828	0.507
$k_1 = 4$	RH–TS	1.000	1.000	1.000	1.000	0.833	0.522
	RH–VNS	1.000	1.000	1.000	1.000	0.965	0.593
	TS–VNS	1.000	1.000	1.000	1.000	0.806	0.540
$k_1 = 5$	RH–TS	1.000	1.000	1.000	0.979	0.979	0.846
	RH–VNS	1.000	1.000	1.000	1.000	0.926	0.784
	TS–VNS	1.000	1.000	1.000	0.979	1.000	0.760
$k_1 = 6$	RH–TS	1.000	1.000	0.951	1.000	0.889	0.588
	RH–VNS	1.000	1.000	1.000	1.000	0.884	0.854
	TS–VNS	1.000	1.000	0.951	1.000	0.988	0.712
$k_1 = 7$	RH–TS	1.000	1.000	0.827	0.984	0.948	0.981
	RH–VNS	1.000	1.000	0.835	0.984	0.933	0.836
	TS–VNS	1.000	1.000	0.914	1.000	0.986	0.830

Note. The convention METHOD1–METHOD2 refers to the agreement between the METHOD1 and METHOD2 partitions. For example, RH–TS refers to the agreement of partitions obtained by the RH and TS methods. Abbreviations are explained in the note to Table 1.

Table 6. *Results for UNGA ideological resolutions network—ARI values for countries.*

		$k_2 = 2$	$k_2 = 3$	$k_2 = 4$	$k_2 = 5$	$k_2 = 6$	$k_2 = 7$
$k_1 = 2$	RH–TS	1.000	1.000	1.000	1.000	1.000	1.000
	RH–VNS	1.000	1.000	1.000	1.000	1.000	1.000
	TS–VNS	1.000	1.000	1.000	1.000	1.000	1.000
$k_1 = 3$	RH–TS	1.000	1.000	1.000	1.000	0.885	0.842
	RH–VNS	1.000	1.000	1.000	0.976	1.000	1.000
	TS–VNS	1.000	1.000	1.000	0.976	0.885	0.842
$k_1 = 4$	RH–TS	0.836	1.000	0.978	1.000	0.957	1.000
	RH–VNS	0.647	1.000	1.000	1.000	1.000	1.000
	TS–VNS	0.725	1.000	0.978	1.000	0.957	1.000
$k_1 = 5$	RH–TS	1.000	0.573	1.000	1.000	0.955	0.565
	RH–VNS	0.334	0.557	1.000	1.000	0.955	0.902
	TS–VNS	0.334	0.465	1.000	1.000	1.000	0.576
$k_1 = 6$	RH–TS	1.000	0.489	0.978	1.000	0.979	0.596
	RH–VNS	0.438	0.528	0.935	0.943	0.979	0.596
	TS–VNS	0.438	0.557	0.957	0.943	1.000	1.000
$k_1 = 7$	RH–TS	1.000	1.000	0.679	0.684	0.962	0.938
	RH–VNS	0.514	0.367	0.672	0.709	0.957	0.938
	TS–VNS	0.514	0.367	0.917	0.865	0.811	0.882

Note. The convention METHOD1–METHOD2 refers to the agreement between the METHOD1 and METHOD2 partitions. For example, RH–TS refers to the agreement of partitions obtained by the RH and TS methods. Abbreviations are explained in the note to Table 1.

Table 7. *Results for UNGA ideological resolutions network—ARI values for ideological resolutions.*

		$k_2 = 2$	$k_2 = 3$	$k_2 = 4$	$k_2 = 5$	$k_2 = 6$	$k_2 = 7$
$k_1 = 2$	RH–TS	1.000	1.000	0.855	0.423	0.483	0.350
	RH–VNS	1.000	0.995	0.779	0.409	0.417	0.254
	TS–VNS	1.000	0.995	0.839	0.970	0.458	0.355
$k_1 = 3$	RH–TS	1.000	1.000	0.988	0.953	0.784	0.425
	RH–VNS	1.000	1.000	0.988	0.932	0.611	0.562
	TS–VNS	1.000	1.000	1.000	0.955	0.558	0.431
$k_1 = 4$	RH–TS	1.000	1.000	0.936	1.000	0.678	0.635
	RH–VNS	1.000	0.988	1.000	0.978	1.000	0.673
	TS–VNS	1.000	0.988	0.936	0.978	0.678	0.697
$k_1 = 5$	RH–TS	1.000	1.000	1.000	0.992	0.983	0.822
	RH–VNS	1.000	0.975	1.000	0.992	0.983	0.900
	TS–VNS	1.000	0.975	1.000	1.000	1.000	0.916
$k_1 = 6$	RH–TS	1.000	1.000	0.946	0.961	1.000	0.631
	RH–VNS	1.000	1.000	0.919	0.983	1.000	0.638
	TS–VNS	1.000	1.000	0.973	0.944	1.000	0.989
$k_1 = 7$	RH–TS	1.000	1.000	1.000	0.907	0.868	0.894
	RH–VNS	1.000	0.939	0.940	1.000	0.961	0.887
	TS–VNS	1.000	0.939	0.940	0.907	0.871	0.993

Note. The convention METHOD1–METHOD2 refers to the agreement between the METHOD1 and METHOD2 partitions. For example, RH–TS refers to the agreement of partitions obtained by the RH and TS methods. Abbreviations are explained in the note to Table 1.

a. The partitions obtained by the three methods can exhibit significant differences even when their criterion functions are the same (or nearly the same).
b. There are some instances where the agreement among the partitions of countries is perfect (or nearly perfect), but the partitions of the resolutions are substantially different. The reverse is also true.
c. Agreement tends to deteriorate as k_1 and k_2 increase, but this is not always the case.

To expound on point (a), consider the ($k_1 = 6, k_2 = 7$) results for the military resolutions network. The criterion functions for the RH, TS, and VNS algorithms were 1,613, 1,608, and 1,599, respectively, as shown in Table 2. Although these variations appear reasonably modest, Tables 4 and 5 reveal that the ARI results reflect some fairly severe differences in the partitions of the countries and resolutions. For the countries, the ARI values for RH–TS, RH–VNS, and TS–VNS comparisons were 0.729, 0.902, and 0.782, respectively. For the military resolutions, the ARI values for RH–TS, RH–VNS, and TS–VNS comparisons were 0.588, 0.854, and 0.712, respectively. Interestingly, the RH and VNS partitions exhibited the greatest similarity, despite the fact that their criterion function values were farthest apart. Indeed, similarity in criterion function values cannot be assumed to reflect similarity in the partitions. Consider, for example, the ($k_1 = 6, k_2 = 3$) results for the military resolutions network, where RH, TS, and VNS each obtained the same criterion function value of 1,805. Table 4 reveals that, despite the equal criterion function values, the partitions of the countries obtained by the three methods were drastically different, with ARI values of 0.469, 0.594, and 0.624 for the RH–TS, RH–VNS, and TS–VNS comparisons, respectively. This is a reflection of the potential for there to be multiple equally well-fitting partitions (see Doreian et al., 2005), where partitions with the same criterion function value can be quite different in some instances.[6]

The ($k_1 = 6, k_2 = 3$) blockmodel for the military resolutions network also provides an excellent example of point (b). Although the partitions of the countries are markedly different (ARI values of 0.469, 0.594, and 0.624 as shown in Table 4), the RH, TS, and VNS partitions of the military resolutions are identical (all ARI values are 1.0 among pairs of military resolutions among the methods as shown in Table 5). This shows that, for a given partition of objects in one mode, there can be considerable flexibility in the cluster assignment of objects in the other mode while maintaining the same criterion function.

Finally, with respect to point (c), the results in Tables 4–7 indicate that there was a strong tendency for (nearly-) identical partitions of the countries and resolutions when k_1 and/or k_2 were small, with greater departures observed when $k_1 > 4$ and $k_2 > 4$. To illustrate, consider the nine combinations stemming from $2 \leqslant k_1 \leqslant 4$ and $2 \leqslant k_2 \leqslant 4$. For the military resolutions network, the country ARI values among all pairs of methods exceeded 0.9 for eight out of nine combinations, while the resolution ARI value among all pairs of methods was 1.0 for all nine combinations. By contrast, consider the nine combinations stemming from $5 \leqslant k_1 \leqslant 7$ and $5 \leqslant k_2 \leqslant 7$. For the military resolutions network, the country ARI values among

[6] With multiple equally well-fitting partitions, there is no principled way of selecting from them. Further, even when the blockmodel structure is the same, this causes problems in interpreting the results.

all pairs of methods exceeded 0.9 for four out of nine combinations, while the resolution ARI values among all pairs of methods exceeded 0.9 for five out of nine combinations. Nevertheless, there were instances where excellent partition agreement among the methods was observed for large values of k_1 and k_2. A good example is the $(k_1 = 6, k_2 = 5)$ results for the military resolutions network, where each method yielded exactly the same partition of countries and exactly the same partition of resolutions.

5.4 Comparison with Pajek

We compared the results of the VNS algorithm for the military resolutions network to those obtained using Pajek for several different combinations of k_1 and k_2. The Pajek software system uses the RH procedure devised by Doreian et al. (2004) to obtain blockmodels based on structural equivalence. Pajek yielded the same unique partition as our RH, TS, and VNS algorithms for the $(k_1 = k_2 = 2)$ blockmodel and, therefore, the resulting ARI was 1.0. For the $(k_1 = k_2 = 3)$ blockmodel, Pajek produced a partition with a criterion function of 1,830, compared to 1,829 for our RH, TS, and VNS implementations, and the resulting ARI was 0.95.[7] At $(k_1 = k_2 = 4)$, the Pajek partition produced 1,749 inconsistencies compared to 1,743 for our RH, TS, and VNS algorithms, and the ARI was 0.91. The Pajek algorithm yielded a drastically inferior blockmodel for $(k_1 = k_2 = 5)$, with a criterion function of 1,706. By contrast, each of the RH, TS, and VNS algorithms produced a partition with a criterion function of 1,663, and its agreement with the Pajek partition was only 0.66 as measured by the ARI.

6 Summary and conclusion

6.1 Summary

We have developed an adaptation of VNS for deterministic two-mode blockmodeling based on structural equivalence. A comparative study across 768 test problems showed that the VNS algorithm significantly outperformed relocation and TS algorithms for the same problem. The VNS algorithm was demonstrated via an application to UNGA voting data, which is an area where two-mode clustering algorithms should hold considerable promise.

6.2 Conclusion

While we used two large UNGA voting data sets, the VNS approach for two-mode blockmodeling would appear to offer considerable promise for other large two-mode binary networks, such as those that might be encountered in supply chain, social media, cognitive diagnosis, and internet purchasing applications. Although other approaches, such as clique partitioning (Grötschel & Wakabayashi, 1989), aggregate

[7] The default option for Pajek software programs computes ARI values after merging the row and column objects. Accordingly, it does not produce separate ARI values for the countries and resolutions independently, but rather a single ARI value considering them jointly as a single set of objects. However, separate row and column values of the ARI measure can be obtained.

information across the resolutions to establish dissimilarity information for subsequent clustering of the countries, the two-mode blockmodeling approach clusters *both* countries and resolutions directly. For this reason, two-mode blockmodeling untangles cleanly the interrelationships between the two modes.

6.3 Limitations and extensions

The RH, TS, and VNS algorithms are not mutually exclusive algorithms. Both the TS and VNS procedures embed the RH and, effectively, reduce to RH under trivial parameter assumptions. For example, the VNS algorithm with $ymax = 0$ is the RH algorithm. Given that TS and VNS embed RH, the nature of RH is important. We used a version of RH with only object transfers and on-the-fly acceptance of improved solutions but many variations are possible. For example, Brusco & Steinley (2011) found that augmenting transfers with exchanges increased computation time, but with no real benefit in solution quality. Another possible strategy that might be pursued is to eliminate the left-bias nature of the algorithm. This can be accomplished by storing all of the row and column object relocations that produce the same best improvement in the criterion function, and then randomly selecting a solution from the tied set. Yet another direction for future research is the development and evaluation of other metaheuristics, perhaps based on simulated annealing, that do not embed the RH algorithm.

Our analyses of the 768 synthetic networks constructed by Brusco & Steinley (2007) revealed that the parameters capturing different network features (e.g., the number of row and column clusters, the relative cluster sizes, and block strength) do make a difference in the relative performance of the heuristic methods. The VNS algorithm substantially outperformed its competitors in this comparative study; however, it is important to acknowledge that different test problem conditions and/or different computation time limits could yield different results. So, although the VNS algorithm appears to be a capable approach for blockmodeling, it should be tested on other blockmodeling problems. One particularly interesting extension would be to test the algorithm for stochastic blockmodeling applications, such as the criterion studied by Karrer & Newman (2011). Those authors used a traditional relocation algorithm in their study; however, we would speculate that VNS would afford an improvement for larger problems with little penalty in terms of computational cost.

References

Batagelj, V., & Mrvar, A. (1998). Pajek—Program for large network analysis. *Connections*, **21**, 47–57.

Batagelj, V., Mrvar, A., Ferligoj, A., & Doreian, P. (2004). Generalized blockmodeling with Pajek. *Metodoloski Zvezki: Journal of the Statistical Society of Slovenia*, **1**, 455–467.

Brusco, M., Doreian, P., Mrvar, A., & Steinley, D. (2013). An exact algorithm for blockmodeling of two-mode network data. *Journal of Mathematical Sociology*, **37**, 61–84.

Brusco, M. J., & Köhn, H.-F. (2009). Clustering qualitative data based on binary equivalence relations: A neighborhood search heuristic for the clique partitioning problem. *Psychometrika*, **74**, 685–703.

Brusco, M., & Steinley, D. (2007). A variable neighborhood search method for generalized blockmodeling of two-mode binary matrices. *Journal of Mathematical Psychology*, **51**, 325–338.

Brusco, M. J., & Steinley, D. (2009). Integer programs for one- and two-mode blockmodeling based on prespecified image matrices for structural and regular equivalence. *Journal of Mathematical Psychology*, **53**, 577–585.

Brusco, M. J., & Steinley, D. (2011). A tabu search heuristic for deterministic two-mode blockmodeling of binary network matrices. *Psychometrika*, **76**, 612–633.

Clinton, J., Jackman, S., & Rivers, D. (2004). The statistical analysis of roll call data. *American Political Science Review*, **98**, 1–16.

Davis, A., Gardner, B., & Gardner, M. R. (1941). *Deep south*. Chicago: University of Chicago Press.

Doreian, P., Batagelj, V., & Ferligoj, A. (2004). Generalized blockmodeling of two-mode network data. *Social Networks*, **26**, 29–53.

Doreian, P., Batagelj, V., & Ferligoj, A. (2005). *Generalized blockmodeling*. Cambridge, UK: Cambridge University Press.

Doreian, P., Lloyd, P., & Mrvar, M. (2013). Partitioning large signed two-mode networks: Problems and prospects. *Social Networks,* **35**, 178–203.

Galaskiewicz, J. (1985). *Social organization of an urban grants economy*. New York: Academic Press.

Grötschel, M., & Wakabayashi, Y. (1989). A cutting plane algorithm for a clustering problem. *Mathematical Programming*, **45**, 59–96.

Grötschel, M., & Wakabayashi, Y. (1990). Facets of the clique partitioning polytope. *Mathematical Programming*, **47**, 367–387.

Hansen, P., & Mladenović, N. (1997). Variable neighborhood search for the *p*-median. *Location Science*, **5**, 207–226.

Hansen, P., & Mladenović, N. (2001). J-Means: A new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, **34**, 405–413.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, **2**, 193–218.

Karrer, B., & Newman, M. E. J. (2011). Stochastic blockmodels and community structure in networks. *Physical Review* E, **83**, 016107(1–10). doi:10.1103/PhysRevE.83.01607.

Kim, S. Y., & Russett, B. (1996). The new politics of voting alignments in the United Nations General Assembly. *International Organization*, **50**, 629–652.

Latapy, M., Magnien, C., & Del Vecchio, N. (2008). Basic notions for the analysis of large two-mode networks, *Social Networks*, **30**, 31–48.

Lorrain, F., & White, H. C. (1971). Structural equivalence of individuals in social networks. *Journal of Mathematical Sociology*, **1**, 49–80.

Madeira, S. C., & Oliveira, A. L. (2004). Biclustering algorithms for biological data analysis: A survey. *IEEE Transactions in Computational Biology and Bioinformatics*, **1**, 24–45.

Mirkin, B., Arabie, P., & Hubert, L. J. (1995). Additive two-mode clustering: The error-variance approach revisited. *Journal of Classification*, **12**, 243–263.

Mische, A., & Pattison, P. (2000). Composing a civic arena: Publics, projects, and social settings. *Poetics*, **27**, 163–194.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, **24**, 1097–1100.

Poole, K. T. (2005). *Spatial models of parliamentary voting*. Cambridge, UK: Cambridge University Press.

Poole, K. T., Lewis, J., Lo, J., & Carroll, R. (2011). Scaling roll call votes with wnominate in R. *Journal of Statistical Software*, **42**(14), 1–21.

Poole, K. T., & Rosenthal, H. (1985). A spatial model for legislative roll call analysis. *American Journal of Political Science*, **29**, 357–384.

Prelić, A., Blueler, S., Zimmermann, P., Wille, A., Bühlmann, P., Gruissem, W.,... Zitzler, E. (2006). A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, **22**, 1122–1129.

Protti, F., Dantas da Silva, M., & Szwarcfiter, J. L. (2009). Applying modular decomposition to parameterized cluster editing problems. *Theory of Computer Systems*, **44**, 91–104.

Règnier, S. (1965). Sur quelques aspects mathématiques des problèmes de classification automatique. *I.C.C. Bulletin*, **4**, 175–191.

Schepers, J., & Van Mechelen, I. (2011). A two-mode clustering method to capture the nature of the dominant interaction pattern in large profile data matrices. *Psychological Methods*, **16**, 361–371.

Selim, H. M., Askin, R. G., & Vakharia, A. J. (1998). Cell formation in group technology: Review, evaluation and directions for future research. *Computers and Industrial Engineering*, **34**, 3–20.

Steinley, D. (2004). Properties of the Hubert-Arabie adjusted Rand index. *Psychological Methods*, **9**, 386–396.

van Mechelen, I., Bock, H. H., & DeBoeck, P. (2004). Two-mode clustering methods: A structured overview. *Statistical Methods in Medical Research*, **13**, 363–394.

van Rosmalen, J., Groenen, P. J. F., Trejos, J., & Castillo, W. (2009). Optimization strategies for two-mode partitioning. *Journal of Classification*, **26**, 155–181.

van Uitert, M., Meuleman, W., & Wessels, L. (2008). Biclustering sparse binary genomic data. *Journal of Computational Biology*, **15**, 1329–1345.

Voeten, E. (2000). Clashes in the assembly. *International Organization*, **54**, 185–217.

Wasserman, S., & Faust, K. (1994). *Social network analysis: Methods and applications.* Cambridge: Cambridge University Press.

Wilderjans, T. F., Depril, D., & Van Mechelen, I. (2013). Additive biclustering: A comparison of one new and two existing ALS algorithms. *Journal of Classification*, **30**, 56–74.