Social Networks 41 (2015) 26–35

Contents lists available at ScienceDirect

Social Networks

journal homepage: www.elsevier.com/locate/socnet

A real-coded genetic algorithm for two-mode *KL*-means partitioning with application to homogeneity blockmodeling

Michael Brusco[a,*], Patrick Doreian[b]

[a] Florida State University, United States
[b] University of Ljubljana, Slovenia and University of Pittsburgh, United States

ARTICLE INFO

Keywords:
Blockmodeling
Clustering
Genetic algorithm
Homogeneity blockmodeling
Real coding
Two-mode *KL*-means partitioning

ABSTRACT

The two-mode *KL*-means partitioning (TMKLMP) problem has a number of important applications in the social and physical sciences. For example, the intra-block variability measure associated with TMKLMP underscores its direct relevance to two-mode homogeneity blockmodeling of binary and real-valued social networks. We present a real-coded genetic algorithm for obtaining TMKLMP solutions. A simulation study showed that the new algorithm compares favorably to a multistart implementation of a two-mode *KL*-means heuristic, which is recognized as a top-performing method for TMKLMP. The merit of the proposed method is demonstrated via an application to the blockmodeling of social network data associated with signing of environmental advertisements in the New York Times as a part of the Turning Point Project.

1. Introduction

Consider a typical data matrix where the rows of the matrix correspond to *n* respondents and the columns correspond to *m* variables on which those respondents are measured. The data consist of measurements for each respondent on each variable. A common clustering approach for such data is to establish a measure of proximity (e.g., squared Euclidean distance) between each pair of respondents using their respective variable measurements and, subsequently, cluster the respondents based on those proximity measures. This is a *one-mode* clustering problem because only the respondents are clustered. The information in the variables is collapsed to establish the proximity measures for the respondents, but the variables are not clustered. A *two-mode* clustering problem is one that would require the establishment of a clustering solution for both the respondents and the variables.[1]

Applications of two-mode clustering abound in the blockmodeling of social network data (Borgatti and Everett, 1997; Brusco, 2011; Brusco et al., 2013a, 2013b, 2013c; Brusco and Steinley, 2007b, 2011; Doreian et al., 2004, 2005, 2013; Everett and Borgatti, 2013; Latapy et al., 2008). Two-mode clustering problems also arise in several other scientific domains, such as the clustering of gene expression data in the biological sciences (Madeira and Oliveira, 2004; Prelić et al., 2006; van Uitert et al., 2008) and part-machine grouping problems in industrial engineering (Selim et al., 1998). The many different formulations of two-mode clustering problems contrast in ways largely based on the nature of the application. Although the complexity of two-mode clustering is apt to vary across different objective criteria and constraints, Madeira and Oliveira (2004, p. 26) observed "...almost all interesting variants of this problem are NP-complete." Excellent surveys of two-mode clustering formulations and algorithms are provided by Madeira and Oliveira (2004) and van Mechelen et al. (2004).

Here, we focus on a two-mode generalization of minimum sum-of-squares clustering, which is a one-mode problem commonly known as *K*-means clustering (Steinhaus, 1956; Forgy, 1965; MacQueen, 1967). Although less well-known, there is a two-mode extension of *K*-means clustering that requires the simultaneous partitioning of two distinct sets of objects (Baier et al., 1997; Brusco and Doreian, in press; Brusco and Steinley, 2007b; Gaul and Schader, 1996; van Rosmalen et al., 2009; Vichi, 2001). Throughout the remainder of this paper, we focus on this particular

* Corresponding author at: Department of Marketing College of Business, Florida State University, 821 Academic Way, Tallahassee, FL 32306-1110, United States.
E-mail addresses: mbrusco@fsu.edu, mbrusco@cob.fsu.edu (M. Brusco), pitpat@pitt.edu (P. Doreian).

[1] More generally, two-mode clustering requires the assignment of two distinct sets of objects to clusters. Each set is a 'mode' of the data, and the number of clusters for each mode need not be the same.

http://dx.doi.org/10.1016/j.socnet.2014.11.007
0378-8733/© 2014 Elsevier B.V. All rights reserved.

generalization of K-means clustering, which we hereafter refer to as the two-mode KL-means partitioning (TMKLMP) problem.[2]

The objective criterion of TMKLMP is a homogeneity measure that is defined as the sum-of-squared deviations of matrix elements from the means associated with blocks (submatrices) of elements formed by the intersection of the clusters for the two sets of objects. This type of intra-block variance measure for two-mode clustering dates back (at least) to the work of Hartigan (1972), with subsequent important contributions in the statistical literature by DeSarbo (1982) and Both and Gaul (1985, 1987). Within the context of social network analysis, it has been recognized that intra-block variance homogeneity measures can be used for the blockmodeling of both real-valued and binary networks based on structural equivalence (Borgatti and Everett, 1992; Brusco and Steinley, 2007b; Žiberna, 2007). Although well-suited for structural equivalence, simulation results reported by Žiberna (2009) reveal that intra-block variance measures also perform well for blockmodeling based on regular equivalence.

Since the mid-1990s, a number of heuristic algorithms have been proposed for TMKLMP. These algorithms include alternating least squares (Gaul and Schader, 1996), two-mode K-means (Baier et al., 1997; Vichi, 2001), simulated annealing (Trejos and Castillo, 2000), genetic algorithms (Hansohm, 2002), tabu search (Castillo and Trejos, 2002), variable neighborhood search (Brusco and Steinley, 2007b) and fuzzy steps (van Rosmalen et al., 2009). More recently, an exact solution procedure was devised by Brusco and Doreian (in press). However, its application is limited to small matrices (20 or fewer row/column objects).

The most comprehensive comparison of methods to date was performed by van Rosmalen et al. (2009, p. 179), who concluded "...the best average performance is obtained using the two-mode K-means method...". The exceptional performance of two-mode K-means clustering in the van Rosmalen et al. (2009) TMKLMP study is concordant with results for one-mode K-means clustering in a study conducted by Brusco and Steinley (2007a). For example, both comparative studies showed the superiority of K-means to implementations of tabu search and simulated annealing. However, Brusco and Steinley (2007a) also found that a genetic algorithm performed slightly better than K-means. In light of this finding, it seems reasonable to contemplate the design of a genetic algorithm for the TMKLMP and a comparison of its performance to two-mode K-means clustering. An earlier investigation of this possibility was conducted by Hansohm (2002), who proposed an *integer-coded* genetic algorithm for TMKLMP. The integer coding corresponds to the fact that the chromosomes in the population were vectors of cluster assignments for the two modes, and each gene of the chromosome was an integer value indicating the cluster to which each object was assigned. As noted by van Rosmalen et al. (2009), the integer-coded genetic algorithm was much less effective relative to its performance in the one-mode context.

Here, we present a *real-coded* genetic algorithm for the TMKLMP wherein the chromosomes of the algorithm are the 'real-valued' centroids of solutions to the TMKLMP. Our selection of this approach is based on three notions: (1) the genetic algorithm that performed so well in Brusco and Steinley's (2007a) one-mode K-means comparative study was a real-coded genetic algorithm; (2) with the real-coding approach, the chromosome vectors are much shorter in length and cluster center information is better preserved in the crossover operation that produces 'offspring' chromosomes from splicing two parent chromosomes; and (3) the two-mode K-means clustering algorithm can efficiently and effectively refine the offspring chromosome.

Section 2 provides a formal presentation of the TMKLMP and describes the real-coded genetic algorithm for its solution. Computational results for the proposed method are reported in Section 3. An empirical application related to two-mode homogeneity blockmodeling is presented in Section 4. The paper concludes in Section 5 with a summary and a discussion of possible extensions.

2. Two-mode *KL*-means partitioning

2.1. Formulation

The notation for the presentation of TMKLMP is provided in Table 1. The optimization problem associated with TMKLMP (Baier et al., 1997; Brusco and Steinley, 2007b; Gaul and Schader, 1996; Hansohm, 2002; Trejos and Castillo, 2000; van Rosmalen et al., 2009; Vichi, 2001) is to find the partitions π and ω that minimize the total sum-of-squared error variation[3] across all KL blocks, which is formally stated as follows:

$$\min_{\pi \in \Pi, \omega \in \Omega} : f(\pi, \omega) = \sum_{k=1}^{K} \sum_{l=1}^{L} v_{kl}. \qquad (1)$$

The minimization of the total sum-of-squared error variation across the blocks is equivalent to maximizing the total variation-accounted-for (VAF) in $\mathbf{X}$:

$$VAF = vaf(\pi, \omega) = \frac{(v - f(\pi, \omega))}{v} \qquad (2)$$

The total number of partitions in Π and Ω are Stirling numbers of the second kind (Clapham, 1996). Since any partition of the mode 1 objects can be matched with any partition of the mode 2 objects, the solution space for TMKLMP is the product of these two Stirling numbers. Essentially, the implications of this result are that the solution space grows exponentially as a function of n and m.

2.2. A two-mode KL-means heuristic

A straightforward heuristic strategy for TMKLMP is based on the extension of K-means clustering heuristics (Forgy, 1965; Steinhaus, 1956). A two-mode adaptation of K-means clustering is described by Baier et al. (1997) as well as several other sources (Vichi, 2001; Brusco and Steinley, 2007b; van Rosmalen et al., 2009). The two-mode KL-means heuristic (TMKLMH) consists of the following steps:

Step TMKLMH0. Construct random initial partitions, π and ω, for the mode 1 and mode 2 object sets respectively, and compute $f^* = f(\pi,\omega)$.
Step TMKLMH1. Reassignment of Mode 1 Objects.
Step TMKLMH1a: Compute $\bar{x}_{kl}$ for all $1 \le k \le K$ and $1 \le l \le L$.
Step TMKLMH1b. Compute $\alpha_{ik} = \sum_{l=1}^{L} \sum_{j \in T_l} (x_{ij} - \bar{x}_{kl})^2$ $1 \le i \le n$ and $1 \le k \le K$.
Step TMKLMH1c. Update π by setting $i \in S_k$: $\alpha_{ik} = \min_{1 \le h \le K} \{\alpha_{ih}\}$, for all $1 \le i \le n$.
Step TMKLMH1d. If $S_k = \varnothing$ for any k $(1 \le k \le K)$, then set $i' \in S_k$: $\alpha_{i'} = \max_{1 \le i \le n} \{ \min_{1 \le h \le K} \{\alpha_{ih}\}\}$. Set $\alpha_{i'k} = 0$ and repeat this step as needed to ensure no empty clusters.

[2] Some authors (Brusco and Steinley, 2007b; van Rosmalen et al. 2009) have employed the term 'two-mode K-means' to refer to the same problem. We adopt the term 'two-mode KL-means' recently used by Brusco and Doreian (in press) which explicitly reflects the fact that the number of clusters for the first mode (K) need not be the same as the number of clusters for the second mode (L).

[3] The use of sum-of-squares error variation in two-mode clustering was originally suggested by Hartigan (1972), who sought to minimize this criterion using a divisive hierarchical algorithm for splitting the dataset into a pre-specified number of blocks.

Table 1
Notation.

n = the number of objects for mode 1 (the rows);
m = the number of objects for mode 2 (the columns);
$\mathbf{X}$ = a $\Re^{n\times m}$ two-mode matrix;
K = the number of clusters for mode 1;
L = the number of clusters for mode 2;
Π = the set of all partitions of the mode 1 objects into K clusters;
π = a partition, $(\pi = \{S_1,\ldots,S_K\}) \in \Pi$, of the mode 1 objects into K clusters, where S_k is the set of objects assigned to cluster k and $n_k = \|S_k\|$ is the number of objects in S_k, for all $1 \le k \le K$;
Ω = the set of all partitions of the mode 2 objects into L clusters;
ω = a partition, $(\omega = \{T_1,\ldots,T_L\}) \in \Omega$, of the mode 2 objects into L clusters, where T_l is the set of objects assigned to cluster l and $m_l = \|T_l\|$ is the number of objects in T_l, for all $1 \le l \le L$;
$\bar{x}$ = the grand mean of $\mathbf{X}$, $\bar{x} = \sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij}/nm$;
υ = the total (sum-of-squares) variation in $\mathbf{X}$, $\upsilon = \sum_{i=1}^{n}\sum_{j=1}^{m}(x_{ij}-\bar{x})^2$
$\bar{x}_{kl}$ = the mean of the elements in the block (*) formed by the mode 1 objects in cluster S_k and the mode 2 objects in cluster T_l, $\bar{x}_{kl} = \sum_{i\in S_k}\sum_{j\in T_l} x_{ij}/n_k m_l$, for all $1 \le k \le K$ and $1 \le l \le L$;
υ_{kl} = the intra-block sum-of-squared error variation in the block formed by the mode 1 objects in cluster S_k and the mode 2 objects in cluster T_l, $\upsilon_{kl} = \sum_{i\in S_k}\sum_{j\in T_l}(x_{ij}-\bar{x}_{kl})^2$, for all $1 \le k \le K$ and $1 \le l \le L$.

* A ‘block’ is, essentially, the submatrix of $\mathbf{X}$ formed by the rows associated with the mode 1 objects in S_k and the columns corresponding to the mode 2 objects in T_l. The term comes from the blockmodeling literature (see Doreian et al., 2005).

Step TMKLMH2. Reassignment of Mode 2 Objects.
Step TMKLMH2a: Compute $\bar{x}_{kl}$ for all $1 \le k \le K$ and $1 \le l \le L$.
Step TMKLMH2b. Compute $\beta_{jl} = \sum_{k=1}^{K}\sum_{i\in S_k}(x_{ij}-\bar{x}_{kl})^2$ $1 \le j \le m$ and $1 \le l \le L$.
Step TMKLMH2c. Update ω by setting $j \in T_l$: $\beta_{jl} = \min_{1\le h\le L}\{\beta_{jh}\}$, for all $1 \le j \le m$.
Step TMKLMH2d. If $T_l = \varnothing$ for any l $(1 \le l \le L)$, then set $j' \in T_l$: $\beta_{ji} = \max_{1\le j\le m}\{\min_{1\le h\le L}\{\beta_{jh}\}\}$. Set $\beta_{j'l} = 0$ and repeat this step as needed to ensure no empty clusters.
Step TMKLMH3. Compute $f(\pi,\omega)$. If $f(\pi,\omega) < f^*$, then set $f^* = f(\pi,\omega)$ and return to Step TMKLMH1; otherwise, stop.

Defining an *iteration* as a cycle through Steps TMKLMH1 and TMKLMH2, the computational requirement per iteration is for $2KL$ centroids, and the evaluation on $nK + mL$ possible assignments of objects to clusters. The number of iterations required for convergence tends to depend on problem size (i.e., n, m, K, and L) and the quality of the initial random partition obtained in Step TMKLMH0.

The performance of TMKLMH is sensitive to the initial partitions obtained in Step TMKLMH0 and, therefore, we recommend the heuristic be restarted many times (e.g., 500 restarts in the previous studies by Brusco and Steinley (2007b) and van Rosmalen et al. (2009)) to avoid the potential for a poor local minimum. Another important concern is that empty clusters can arise in Steps TMKLMH1c and TMKLMH2c and, accordingly, this is remedied by reassignment of the case that is farthest from its current cluster centroid to the empty cluster (see Steps TMKLMH1d and TMKLMH2d).

In an extensive comparative study reported by van Rosmalen et al. (2009), a multiple restart (multistart) implementation of TMKLMH generally outperformed all of the following: an exchange procedure (Gaul and Schader, 1996); a simulated annealing heuristic (Trejos and Castillo, 2000); and a tabu search method (Castillo and Trejos, 2002). Thus, multistart TMKLMH is considered one of the best available methods for TMKLMP. Nevertheless, computational results for one-mode K-means clustering (see Brusco and Steinley, 2007a) suggest that embedding K-means heuristics within a real-coded genetic algorithm can yield better performance than multiple restarts alone. Accordingly, in the next subsection, we propose a new real-coded genetic algorithm for TMKLMP adopting this strategy.

2.3. A real-coded genetic algorithm

An initial population is obtained by applying restarts of TMKLMH, obtaining the $\bar{x}_{kl}$ values for each solution, and unfolding these centroids into KL length vectors representing the chromosomes. Thus, each unique $\bar{x}_{kl}$ value is a gene of the chromosome. Offspring centroids are then obtained by crossover operations that splice the chromosomes of two randomly-selected ‘parents’ from the population, as well as mutation operations that occasionally perturb some of the genes in the chromosomes. The precise steps of our implementation of the two-mode genetic algorithm (TMGA) are as follows:

Step TMGA0. Establish a population of C chromosomes by running $C_{\max}$ restarts of TMKLMH and unfolding the centroids (i.e., the final $\bar{x}_{kl}$ values) of the C best restarts (i.e., those restarts producing the C smallest $f(\pi,\omega)$ values) into KL length real-valued vectors. Define $\mathbf{P}$ as the $KL \times C$ matrix with columns corresponding to these chromosomes. Let π^* and ω^* correspond to the partitions yielding the minimum value of $f(\pi,\omega)$ across all of the restarts. Set the mutation probability parameter (λ), the maximum number of iterations with no improvement parameter ($\tau_{\max}$), and its counter $\tau = 0$.

Step TMGA1. Crossover (chromosome splicing).
Step TMGA1a: Randomly select two vectors, $\mathbf{p}_1$ and $\mathbf{p}_2$, from $\mathbf{P}$.
Step TMGA1b. Randomly select an integer, d, on the interval $[2, KL-1]$.
Step TMGA1c. Create a new chromosome, $\mathbf{q}$, by splicing the first d elements of $\mathbf{p}_1$ with the last KL-d elements of $\mathbf{p}_2$.
Step TMGA2. Mutation.
Step TMGA2a: Generate a KL-length vector, $\mathbf{u}$, of uniform [0,1] random numbers.
Step TMGA2b. If $u_h < \lambda$, then replace q_h with a uniformly-distributed random number on the interval bounded by the minimum and maximum values in $\mathbf{X}$.
Step TMGA3. Evaluation
Step TMGA3a. Fold $\mathbf{q}$ into the $\bar{x}_{kl}$ values.
Step TMGA3b. Using $\bar{x}_{kl}$ as input, apply steps TMKLMH1 and TMKLMH2 of the TMKLMH procedure to reassign cases and obtain π and ω.
Step TMGA3c. Compute $f(\pi,\omega)$. If $f(\pi,\omega) < f(\pi^*,\omega^*)$, then set $\pi^* = \pi$, $\omega^* = \omega$, $\tau = 0$, update $\mathbf{P}$ by replacing the chromosome corresponding to the largest value of the objective function with the $\bar{x}_{kl}$ values for

π^* and ω^*, and return to Step TMGA1; otherwise, proceed to Step TMGA3d.
Step TMGA3d. Set $\tau = \tau + 1$. If $\tau > \tau_{\max}$, then stop; otherwise, return to step TMGA1.

Defining an *iteration* as a cycle through Steps TMGA1, TMGA2 and TMGA3, the computational requirement for the algorithm consists of random selection of two chromosomes to form a *KL*-length vector, a random perturbation operation on the *KL*-length vector, and then iterations of the TMKLMH algorithm until convergence. As noted in the previous subsection, computational requirement for TMKLMH is for 2*KL* centroids, and evaluation of $nK + mL$ possible assignments of objects to clusters. Accordingly, the run-time requirement for an iteration of TMGA is not much different from TKMLMH, and fewer iterations are often required for convergence because the initial partition is not random but obtained from splicing together two sets of centroids.

In Step TMGA0, $C_{\max}$ restarts of the TMKLMH algorithm are used to establish an initial population of C chromosomes, and the mutation probability and termination parameters are initialized. So, for example, if $C_{\max} = 1000$ and $C = 100$, then 1000 restarts of the TMKLMH are applied and the initial population of chromosomes corresponds to the 100 best solutions (i.e., smallest $f(\pi,\omega)$ values) across those 1000 restarts. A crossover operation is used in Step TMGA1 to produce a new 'offspring' chromosome. Each gene in the new chromosome is mutated with probability λ in Step TMGA2. If a gene is randomly selected for mutation, then this is accomplished by randomly selecting a real value on the range of the data. If a new best-found solution is identified in Step TMGA3, then it is installed and the index τ is reset to zero. The population of chromosomes (**P**) is also updated each time a new best-found solution is realized in Step TMGA3c. If a new best-found solution is not obtained, then τ is incremented, and the algorithm terminates if $\tau > \tau_{\max}$.

3. Computational results

3.1. Experimental design

We adopt the experimental design used by van Rosmalen et al. (2009) in their comparative evaluation of methods for TMKLMP. The authors manipulated four design features at three levels each. The first design feature was the number of objects for mode 1 and mode 2, which was tested at levels of ($n = 60$, $m = 60$), ($n = 120$, $m = 120$), and ($n = 150$, $m = 30$). The second design feature, the number of clusters for modes 1 and 2, was tested at levels of ($K = L = 3$), ($K = L = 5$), and ($K = L = 7$). Cluster density, which reflects the relative sizes of the mode 1 and mode 2 clusters, was the third design feature. The first level of cluster density assumed clusters of equal size for the two sets of objects. The second level assumed, for each mode, that there was one large cluster consisting of 60% of the objects and that the remaining objects were equally distributed among the remaining clusters. The third level assumed, for each mode, that there was one small cluster consisting of 10% of the objects and that the remaining objects were equally distributed among the remaining clusters. The fourth design feature was the standard deviation (σ) of the normally distributed random error introduced to the data, which was tested at levels of $\sigma = .5$, $\sigma = 1.0$, and $\sigma = 2.0$.

The four design features, each with three levels, yield a total of $3^4 = 81$ unique test data configurations. For each configuration, n, K, and the density level were used to establish an $n \times K$ matrix (**Y**) of mode 1 cluster assignments. Similarly, m, L and the density level were used to produce an $m \times L$ matrix (**Q**) of mode 2 cluster assignments. A $K \times L$ centroid matrix, **W**, was established by randomly assigning values of the inverse standard normal cumulative distribution function, $\Phi(h/(KL + 1))$, for all $1 \le h \le KL$, to the elements of **W**. Finally, elements of an $n \times m$ error matrix, **E**, were randomly generated from a normal distribution with mean zero and standard deviation σ. The data matrix, **X**, for the configuration was then constructed using **Y**, **Q**, **W**, and **E** as follows:

$$\mathbf{X} = \mathbf{YWQ}' + \mathbf{E}. \quad (3)$$

The data were generated using MATLAB. Ten replications for each of the 81 design configurations were generated by changing the random number generation seed, which resulted in 810 distinct datasets.

3.2. Implementation of methods

The TMKLMH and TMGA algorithms were programmed as MATLAB m-files and are freely available from the first author. We acknowledge that the selection of the number of restarts for TMKLMH and TMGA for our simulation experiment has some degree of arbitrariness; however, three pragmatic principles were employed. First, we considered guidelines from two previous studies (Brusco and Steinley, 2007b; van Rosmalen et al., 2009), where 500 restarts were used for TMKLMH. Second, we sought to expand on the number of restarts for TMKLMH, while preserving computational feasibility for the simulation experiment. Third, we adapted the code so that we could measure the best-found *VAF* measure after different levels for the number of restarts. Accordingly, TMKLMH was implemented using 2000 restarts (four times the number in previous studies); however, intermediate results were collected after 10, 100, 500, and 1000 restarts.

Two different versions of the TMGA algorithm were considered. The first version, TMGA(a), was designed to measure the benefit of genetic search above and beyond what can be achieved with TMKLMH. The parameter settings for TMGA(a) were $C_{\max} = 2000$, $C = 100$, $\tau_{\max} = 2000$. Because TMGA(a) uses the same number of restarts as TMKLMH, its performance must be at least as good as TMKLMH on each test problem. The second version, TMGA(b), was designed to be a direct competitor with TMKLMH in terms of computational effort, with parameter settings of $C_{\max} = 1000$, $C = 100$, $\tau_{\max} = 1000$, and $\lambda = .05$. Accordingly, TMGA(b) is allowed only one-half the number of restarts as TMKLMH; however, it is also permitted genetic search iterations that attempt to escape from local optima via crossover and mutation operations. For TMGA(b), we collected intermediary results for the genetic search process after 10, 100, 500, and 1000 search iterations.

The TMKLMH, TMGA(a), and TMGA(b) algorithms were applied to each of the 810 test problems, and four performance measures were stored for each algorithm on each problem: (1) the best-found *VAF* value, (2) the total computation time, (3) the partition recovery for the mode 1 objects, and (4) the partition recovery for the mode 2 objects.[4] Among these measures, *VAF* is the most important because it is the criterion that the methods seek to optimize. Partition recovery, which was measured using the adjusted Rand index (ARI, Hubert and Arabie, 1985), is also an important criterion because it helps to determine whether differences in the algorithms translate into salient differences in their recovery of the partitions of row and column objects based on the data generation process. Throughout the remainder of this paper, we refer to the partitions from the data generation process as the *planted* partitions. The ARI, which is generally recognized as the best standard for measuring partition agreement (Steinley, 2004), achieves a maximum value

[4] The TMKLMH and TMGA algorithms were written as MATLAB m-files, and the simulation study was completed using MATLAB version R2012b(8.0.0.783). The hardware platform was a microcomputer using a 3.4 GHz Intel Core i7-2600 processor with 8.0 GB of RAM. All of the MATLAB files for replicating the simulation study are available from the first author on request.

of 1 for perfect agreement, whereas values near zero suggest only chance agreement.[5]

3.3. Experimental results

Table 2 provides a summary of the average performance measures for the planted partitions and each of the three methods across all 810 test problems. The average computation times for TMKLMH and TMGA(b) were roughly 50 s, whereas TMGA(a) required approximately twice the amount of computation time as its competitors. The TMGA(a) algorithm yielded the best average *VAF* value of .43616, followed by TMGA(b) at .43604 and TMKLMH at .43579. The average *VAF* for the planted partitions was .43095. Denoting *VAF** as the best *VAF* value across the three methods for any given test problem, TMGA(a), TMGA(b), and TMKLMH matched *VAF** for 95.6%, 85.2%, and 77.7% of the test problems, respectively. The planted partitions matched *VAF** for 45.2% of the test problems.

Turning to head-to-head comparisons, TMGA(a) produced a better objective function than TMKLMH for 173 (21.4%) of the 810 test problems. As noted above, TMGA(a) cannot perform worse than TMKLMH because TMKLMH is, effectively, used to establish the initial population for TMGA(a). Our results support the finding of van Rosmalen et al. (2009) that TMKLMH is an excellent performer, but the genetic algorithm was able to improve its result in more than one-fifth of the test instances. The results in Table 2 also reveal that improvement afforded in the *VAF* by TMGA(a) translated into modest improvement in the recovery of the row and column cluster structures. The average ARI values for the mode 1 objects for TMKLMH and TMGA(a) were .828 and .836, respectively. Similarly, the average ARI values for the mode 2 objects for TMKLMH and TMGA(a) were .865 and .868, respectively.

The TMGA(b) implementation affords a more equitable comparison to TMKLMH because of their comparable computation times. Across the 810 test problems, the two algorithms obtained the same *VAF* in 633 (78.1%) instances, TMGA(b) yielded a better *VAF* for 153 (18.9%) problems, and TMKLMH provided a better *VAF* for 24 (3.0%) problems. Both a *t*-test on mean *VAF* performance and a nonparametric sign test indicate that TMGA(b) outperformed TMKLMH (p-value < .01). The TMGA(b) algorithm yielded a larger ARI for mode 1 objects than TMKLMH (.836 vs. .828), but a slightly lower average ARI for mode 2 objects (.864 vs. .865).

There are two important caveats regarding the relationship between the *VAF* and ARI results in Table 2. The first is that meaningfulness of the ARI values is predicated on the quality of the *VAF* associated with the planted partitions. We elaborate on this issue when discussing the detailed results for the design feature levels below. The second caveat pertains to the fact that small *VAF* differences do not always imply small ARI differences. The *VAF* averages range from .43579 to .43616, the mode 1 ARI averages range from .828 to .836, and the mode 2 ARI averages range from .864 to .868. These narrow ranges for the averages can lead to the conclusion that small *VAF* differences translate to small ARI differences; however, closer inspection revealed this is not always the case. Across the 810 test problems, the maximum *VAF* difference between TMGA(a) and TMKLMH was only .00983, and most observed differences were appreciably smaller. Contrastingly, the maximum mode 1 and mode 2 ARI differences between TMGA(a) and TMKLMH were much greater at .449 and .414, respectively. Moreover, there were 33 test problems where the mode 1 ARI difference between the TMGA(a) and TMKLMH solution was .10 or greater, as well 27 instances where the mode 2 ARI difference was .20 or greater. The key point associated with these results is that it would be a mistake to assume that small differences in the *VAF* values always translate into trivial differences in the mode 1 and mode 2 partitions.

Table 3 provides average *VAF* results, for each of the three methods and the planted partitions corresponding to the data generation process, for different levels of each of the four design features. Similarly, Table 4 offers the percentage of test problems for which the best-found *VAF* (*VAF**) was achieved. The most important aspects of these tables are: (1) the TMGA(b) algorithm provided an average *VAF* that was equal to or larger than the average *VAF* of TMKLMH at all levels of all design features, (2) the TMGA(b) algorithm obtained *VAF** for a greater percentage of test problems than TMKLMH at all levels of all design features, (3) the TMGA(a) algorithm provided an average *VAF* that was equal to or larger than the average *VAF* of TMGA(b) at all levels of all design features, (4) the TMGA(a) algorithm obtained *VAF** for a greater percentage of test problems than TMGA(b) at all levels of all design features, and (5) the planted partition produced excellent *VAF* values for the $\sigma = .5$ design feature level; however, the algorithms generally yielded better *VAF*'s than the planted partitions at the higher error level conditions of $\sigma = 1.0$ and $\sigma = 2.0$.

All three methods yielded exceptional performance when there were $K = L = 3$ clusters of mode 1 and mode 2 objects. At this design feature level, each method yielded an average *VAF* of approximately .39098, whereas the average *VAF* for the planted partitions was .38764. The percentages of best-found *VAF* values at $K = L = 3$ for the planted partitions, TMKLMH, TMGA(a), and TMGA(b) were 51.1%, 97.4%, 99.6%, and 98.9%, respectively. Contrastingly, at $K = L = 7$ clusters, there was much greater disparity in the performance of the algorithms, where the average *VAF* values for the planted partitions, TMKLMH, TMGA(a), and TMGA(b) were .46713, .47300, .47399, and .47369, respectively. Likewise, the percentage of best-found *VAF* values at $K = L = 7$ for the planted partitions, TMKLMH, TMGA(a), and TMGA(b) were 43.7%, 56.7%, 91.5%, and 70.7%, respectively.

All three methods also yielded good performances when there was an equal number of objects in each cluster. At this design feature level, the average *VAF* values for the planted partition, TMKLMH, TMGA(a), and TMGA(b) were .44454, .44820, .44838, and .44832, respectively, and the percentage of best-found *VAF* values for TMKLMH, TMGA(a), and TMGA(b) were 87.8%, 99.3%, and 90.7%, respectively. Contrastingly, there was much greater disparity among the methods when there was one small cluster containing 10% of the objects, with an equal distribution of objects among the remaining clusters. At the 10% density level, the average *VAF* values for the planted partitions, TMKLMH, TMGA(a), and TMGA(b) were .41253, .41950, .42202, and .41996, respectively, and the percentage of best-found *VAF* values for the planted partitions, TMKLMH, TMGA(a), and TMGA(b) were 41.1%, 60.4%, 88.9%, and 76.3%, respectively.

All three methods performed well at the low-error design feature level of $\sigma = .5$, where the average *VAF* values for the planted partitions, TMKLMH, TMGA(a), and TMGA(b) were .72996, .72966, .72990, and .72976, respectively, and the percentage of best-found *VAF* values for the planted partition, TMKLMH, TMGA(a), and TMGA(b) were 93.7%, 91.5%, 97.0%, and 95.6%, respectively. Clearly, at this low-error setting, the planted partitions reflect the strong cluster structure, and the algorithms often match, but seldom surpass, the *VAF* of the planted partitions. We probed more deeply into the results for the low-error setting, and discovered that all of the instances where TMGA(a) failed to match the *VAF* of the planted partitions occurred at the 10% density condition. This reinforces the finding observed in the preceding paragraph, which suggests that the 10% density condition is especially challenging for these algorithms.

[5] More precisely, for two partitions: (i) ARI ≥ 0.9 indicates excellent agreements; (ii) 0.9 > ARI ≥ 0.8 suggests a good agreement; (iii) 0.8 > ARI ≥ 0.65 can be viewed as a moderate agreement; and (iv) ARI ≤ 0.65 indicates poor agreements.

Table 2
Simulation results—overall performance summary.

	Planted	TMKLMH	TMGA(a)	TMGA(b)
Average computation time (s)	–	50	102	51
Average *VAF*	.43095	.43579	.43616	.43604
Number of best-found *VAF*'s out of 810 possible	366	629	774	697
Percentage of best-found *VAF*'s out of 810 possible	45.2	77.7	95.6	85.2
......Number of *VAF*'s better than *Planted*	–	438	443	441
Number of *VAF*'s better than TMKLMH	29	–	173	153
Number of *VAF*'s better than TMGA(a)	9	0	–	30
Number of *VAF*'s better than TMGA(b)	16	24	113	–
Average ARI (mode 1 objects)	–	.828	.836	.836
Average ARI (mode 2 objects)	–	.865	.868	.864

Table 3
Simulation results—average *VAF* results by design feature levels.

Design feature	Level	*Planted*	TMKLMH	TMGA(a)	TMGA(b)
Number of objects	$n = m = 60$	.43163	.43705	.43752	.43741
	$n = m = 120$	.43085	.43155	.43169	.43152
	$n = 150$, $m = 30$	.43037	.43876	.43926	.43918
Number of clusters	$K = L = 3$	.38764	.39098	.39098	.39098
	$K = L = 5$	.43807	.44339	.44351	.44345
	$K = L = 7$	.46713	.47300	.47399	.47369
Cluster density	Even distribution	.44454	.44820	.44838	.44832
	60% in largest	.43578	.43967	.43988	.43984
	10% in smallest	.41253	.41950	.42022	.41996
Error level	$\sigma = .5$	.72996	.72966	.72990	.72976
	$\sigma = 1.0$	.41035	.41166	.41181	.41178
	$\sigma = 2.0$	.15254	.16604	.16677	.16658

At the $\sigma = 2.0$ level of error, the average *VAF* values for the planted partition, TMKLMH, TMGA(a), and TMGA(b) were .15254, .16604, .16677, and .16658, respectively, and the percentage of best-found *VAF* values for the planted partitions, TMKLMH, TMGA(a), and TMGA(b) were .7%, 53.0%, 91.5%, and 64.8%, respectively. At this high-error setting, two observations are evident: (i) it is clear that all of the algorithms generally produce a better *VAF* than the planted partitions, and (ii) there is greater variability in performance among the algorithms. Although the average *VAF* for the planted partitions (.15254) is not seriously lower than that of TMGA(a) (.16677) and the other two algorithms, the small percentage of best-found *VAF* values for the planted partitions is of concern when interpreting ARI values. To illustrate, we found that the average ARI values for all three algorithms are in the .98 to .99 range for the $\sigma = .5$ error level, and in the .92 to .95 range for $\sigma = 1.0$. This is excellent recovery and suggests that the planted partitions are a reasonable indicator of underlying cluster structure. However, at the $\sigma = 2.0$ error level, the average ARI values for all three algorithms are in the .59 to .66 range, which leads to the conclusion that recovery of cluster structure cannot legitimately be measured at this highest level of error.

3.4. Evolution of VAF as a function of restarts and search iterations

To complete the analysis of the simulation results, we examine how the solution quality of the THKLMH and TMGA(b) algorithms evolves over time. Concretely, using the number of restarts as a surrogate for time, Table 5 reports the average *VAF* and percentage of best-found *VAF*'s for TMKLMH at 10, 100, 500, 1000, and 2000 restarts. The results for the TMGA(b) heuristic at 1000 restarts and zero genetic search iterations are equivalent to those of TMKLMH

Table 4
Simulation results–Percentage of best-found *VAF* results by design feature levels.

Design Feature	Level	*Planted*	TMKLMH	TMGA(a)	TMGA(b)
Number of objects	$n = m = 60$	44.1	74.8	95.6	82.6
	$n = m = 120$	59.6	87.4	96.7	90.4
	$n = 150$, $m = 30$	31.9	70.7	94.4	82.6
Number of clusters	$K = L = 3$	51.1	97.4	99.6	98.9
	$K = L = 5$	40.7	78.9	95.6	85.9
	$K = L = 7$	43.7	56.7	91.5	70.7
Cluster density	Even distribution	47.8	87.8	99.3	90.7
	60% in largest	46.7	84.8	98.5	88.5
	10% in smallest	41.1	60.4	88.9	76.3
Error level	$\sigma = .5$	93.7	91.5	97.0	95.6
	$\sigma = 1.0$	41.1	88.5	98.1	95.2
	$\sigma = 2.0$	.7	53.0	91.5	64.8

Table 5
Simulation results–Evolution of *VAF* improvement for TMKLMH and TMGA(b).

	TMKLMH Results		TMGA(b) Results	
	Average *VAF*	Percentage of best-found *VAF*'s	Average *VAF*	Percentage of best-found *VAF*'s
10 Restarts	.42886	41.98	–	–
100 Restarts	.43428	64.57	–	–
500 Restarts	.43538	71.98	–	–
1000 Restarts	.43556	74.32	–	–
2000 Restarts	.43579	77.65	–	–
1000 Restarts				
0 genetic search iterations	–	–	.43556	74.32
10 genetic search iterations	–	–	.43570	75.93
100 genetic search iterations	–	–	.43585	78.77
500 genetic search iterations	–	–	.43596	81.11
1000 genetic search iterations	–	–	.43601	83.58
All genetic search iterations	–	–	.43604	85.19

at 1000 restarts, thus providing a balance point for the two algorithms. Table 5 also reports the performance measures (average *VAF* and percentage of best-found *VAF*'s) for TMGA(b) after 10, 100, 500, 1000, and 'all' of the genetic search iterations employed by the algorithm. The 'all' category reflects that the total number of genetic search iterations is not deterministic because the termination criterion is based on $\tau_{\max} = 1000$, which is the number of genetic search iterations *with no improvement* in the objective function. Together, the results in Table 5 provide information regarding (i) how TMKLMH solution quality evolves as a function of the number of restarts, and (ii) the evolution of the improvement in an initial solution provided by TMKLMH as a function of the genetic search iterations employed by TMGA(b).

The TMKLMH heuristic performed reasonably well with only 10 restarts, with an average *VAF* of .42886 and realization of the best-found *VAF* for nearly 42% of the test problems. Increasing the number of restarts to 100 only improved the average *VAF* by approximately .006; however, the number of best-found *VAF* values increased to roughly 65%. Using 500 restarts, as was reported by van Rosmalen et al. (2009), improved average *VAF* to .43538 and nearly 72% of the best-found *VAF* values were obtained. This finding suggests that the 500 restarts used in this earlier study provided a reasonable estimate of average *VAF* for TMKLMH because increasing the number of restarts to 2000 only improved average *VAF* by .00041 to .43579. At 1000 restarts, TMKLMH yielded an average *VAF* of .43556 and matched the best-found *VAF* approximately 74% of the time. Notice that these results are identical to those reported for TMGA(b) with 0 genetic search iterations because TMKLMH with 1000 restarts is the starting point for TMGA(b).

The key result in Table 5 is that TMGA(b) using 1000 restarts and only 100 genetic search iterations outperformed TMKLMH with 2000 restarts with respect to both average *VAF* (.43585 vs. .43579) and percentage of best-found *VAF* values (78.77% vs. 77.65%). This is important because it clearly reveals that the modest computational investment of 100 genetic search iterations using TMGA(b) produced better results than the more demanding burden of 1000 additional restarts (from 1000 to 2000) for TMKLMH. Moreover, 500 genetic search operations for TMGA(b), which is still modest in comparison to 2000 restarts for TMKLMH, yielded improvement of average *VAF* to .43596 and matched the best-found *VAF* for approximately 81% of the test problems. Using 1000 genetic search iterations for TMGA(b) produced an average *VAF* of .43601 and obtained approximately 84% of the best-found *VAF*'s. Once again, it is critical to stress that applying TMGA(b) with 1000 (random) restarts plus 1000 genetic search iterations will require less computation time than using TMKLMH with 2000 (random) restarts because the starting solutions in the genetic search process are much better than random initial solutions and, accordingly, require less time to refine to a local optimum.

4. An example—The Turning Point Project

4.1. The two-mode network

The computational results in the previous section reveal that TMGA generally outperforms TMKLMH with respect to the *VAF* criterion function. Although the *VAF* criterion function differences are typically quite small, as we noted above, this should not be misinterpreted as evidence that the partition differences are necessarily trivial. To illustrate this point, we consider an application of TMKLMP to the blockmodeling of social network data related to the Turning Point Project (TPP).[6] The two modes consist of $n = 108$ organizations that each signed one or more of $m = 25$ radical, environmental activist-oriented[7] advertisements in the New York Times (NYT) during 1999–2000. Different sets of organizations signed different advertisements—but with considerable overlaps for some signing organizations. Accordingly, the data are in the form of a 108×25 two-mode binary matrix (**X**) that corresponds to ties between organizations and full one-page advertisements in the NYT that they signed. The elements of **X** are $x_{ij} = 1$ if organization i signed advertisement j and $x_{ij} = 0$ did not sign the advertisement. The total number of organizations signing a specific advertisement ranged from 16 to 28. The number of signatures from organizations ranged from 1 (18 organizations) to 22 (1 organization). There was great variation in the overall participation of organizations. Going into the clustering analysis, our expectation was that most organizations signing multiple advertisements were more likely to concentrate them into specific substantive domains.

The application of TMKLMP methods to the binary network matrix, **X**, ought to produce partitions of the organizations and advertisements such that the elements of the submatrices produced by crossing one of the organization clusters with one of the advertisement clusters is either mostly 0s or mostly 1s. In the vernacular of social network analysis, such crossings are called blocks, and a block of all (or mostly) zeros is called a null block, whereas a block of all (or mostly) ones is called a complete block. Therefore, in this particular context, the TMKLMP methods are used as a 'two-mode blockmodeling' tool that seeks to establish a $K \times L$ 'image matrix' that is a parsimonious representation of the much larger $n \times m$ network matrix. The elements of the image matrix are the identifiers 'complete' (if the block contains mostly ones) or 'null' (if the block contains mostly zeros). A comprehensive treatment of blockmodeling is provided by Doreian et al. (2005), and

[6] These data were collected and assembled by the second author from whom they are available. A fuller description of the data is provided by Brusco et al. (2013c).
[7] Every advertisement finished with information about, and calls for, action to be taken by readers.

Table 6
The *VAF* results for the Turning Point Project network.

		$L=3$	$L=4$	$L=5$	$L=6$
$K=3$	TMKLMH	.27286858	.28313530	.28934383	.29271834
	TMGA	.27286858	.28313530	.28934383	.29271834
$K=4$	TMKLMH	.32567962	.34416520	.35606338	.35888974
	TMGA	.32567962	.34416520	.35606338	**.35897234**
$K=5$	TMKLMH	.36614905	.39720306	.40837899	.41296559
	TMGA	.36614905	.39720306	.40837899	.41296559
$K=6$	TMKLMH	.38758827	.43958942	.45569737	.45957118
	TMGA	.38758827	.43958942	.45569737	**.46069277**
$K=7$	TMKLMH	.39957810	.46123685	.50177582	.50748522
	TMGA	**.39964065**	.46123685	.50177582	.50748522
$K=8$	TMKLMH	.40983005	.47620763	.52631504	.53354713
	TMGA	**.41102357**	.47620763	.52631504	.53354713
$K=9$	TMKLMH	.41949578	.48702308	.54269325	.55097730
	TMGA	**.41969739**	**.48765685**	**.54269893**	**.55148845**
$K=10$	TMKLMH	.42355529	.49625425	.55587062	.56696906
	TMGA	**.42703290**	**.49677179**	**.55616637**	**.56748022**

Note—The 12 instances where TMGA yielded a better *VAF* than TMKLMH are highlighted in bold.

Table 7
Image matrix for the Turning Point Project network.

TMGA	TMKLMH	GE ads	EG ads	IA ads	TM ads	EC ads
$\boldsymbol{n_1=3}$	$\boldsymbol{n_1=3}$	**Complete**	**Complete**	**Complete**	**Complete**	Null
$\boldsymbol{n_2=4}$	$\boldsymbol{n_2=4}$	**Complete**	**Complete**	Null	Null	**Complete**
$\boldsymbol{n_3=11}$	$\boldsymbol{n_3=11}$	**Complete**	Null	**Complete**	Null	Null
$\boldsymbol{n_4=8}$	$\boldsymbol{n_4=8}$	**Complete**	Null	Null	Null	Null
$\boldsymbol{n_5=10}$	$\boldsymbol{n_5=10}$	Null	**Complete**	Null	Null	Null
$n_6=13$	$n_6=10$	Null	Null	**Complete**	Null	Null
$n_7=10$	$n_7=18$	Null	Null	Null	**Complete**	Null
$\boldsymbol{n_8=21}$	$\boldsymbol{n_8=21}$	Null	Null	Null	Null	**Complete**
$n_9=28$	$n_9=23$	Null	Null	Null	Null	Null

Note—The first two columns contain the number of organizations in each of the 9 clusters for the TMGA and TMKLMH partitions, respectively. Values in bold in the first two columns indicate that the TMGA and TMKLMH cluster memberships were the same. The remaining columns identify the complete and null blocks for the 9×5 image matrix, which was the same for the TMGA and TMKLMH partitions.

contributions related to two-mode blockmodeling are found in Borgatti and Everett (1997), Brusco et al. (2013a, 2013b), Brusco and Steinley (2007b, 2011), Doreian et al. (2004), Doreian et al. (2013) and Everett and Borgatti (2013).

4.2. Results and analysis

The TMKLMH and TMGA (implementation TMGA(b)) algorithms were applied to the TPP network for all 32 combinations of clusters on the intervals of $3 \le K \le 10$ and $3 \le L \le 6$. The *VAF* results are reported in Table 6. Across the 32 combinations, the TMKLMH and TMGA algorithms produced the same *VAF* values for 20 combinations, whereas the TMGA algorithm yielded a better *VAF* in the remaining 12 instances, which are highlighted in bold in the table. Eight of the 12 instances where TMGA provided a solution with a better *VAF* than TMKLMH corresponded to the $K=9$ and $K=10$ conditions, where TMGA was better at each of the four settings for L.[8]

An inspection of the *VAF* values in Table 6 reveals that $L=5$ is a consistently good choice for the number of clusters for the advertisements. For most values of K, there was a marked improvement in *VAF* when going from $L=4$ to $L=5$ clusters, but very little additional improvement when going from $L=5$ to $L=6$ clusters. A selection of $L=5$ clusters is also conceptually appealing[9] because the cluster memberships consistently corresponded to five meaningful categories defined by the organizers of the TPP, which provide insight into the natural or logical partitioning of the advertisements into underlying themes. The categories were: Ecological catastrophe (EC); Economic globalization (EG); Genetic Engineering (GE); Industrial agriculture (IA) and Techno-mania (TM).

A selection of $K=9$ clusters for the organizations was made based on the modesty of the improvement in *VAF* associated with increasing K to 10, as well as the interpretability of the solution. The difference in the *VAF* values for TMKLMH and TMGA at $K=9$ and $L=5$ clusters was extremely modest (*VAF* = .54269325 for TMKLMH vs. *VAF* = .54269893 for TMGA). Despite this seemingly paltry difference of .00000568 in the *VAF* values, there were some noteworthy differences in the partitions of the organizations for the two methods. As shown in the first two columns of Table 7, six of the nine clusters of organizations were the same for the TMKLMH and TMGA methods; however, three were appreciably different in size. As a formal measure of agreement, we computed an adjusted Rand index of .766 between the TMKLMH and TMGA partitions for the organizations. Based on the standards in Steinley (2004), a value of .766 is at the high end of the range for 'fair' agreement between the partitions. Thus, despite the modest difference in *VAF* associated with TMKLMH and TMGA, their partitions of the organizations have some marked differences.

[8] The fact that the superiority of TMGA increases for a larger number of clusters is an important result, which is consistent with the findings for genetic algorithms in the one-mode K-means context (Brusco and Steinley, 2007a). Moreover, it suggests that an expansion of the test conditions in van Rosmalen et al. (2009) to include K/L combinations greater than $K=L=7$ might lead to greater discernment among competitive methods.

[9] That the clustering of the columns corresponds exactly with the five substance areas of the advertisements, despite many organizations signing advertisements in multiple domains, adds to validity of the results from using the methods described here.

Although there are some salient differences in some of the organization clusters associated with the TMKLMH and TMGA solutions, both methods produced the same image matrix, which is displayed in the last five columns of Table 7. From a blockmodeling perspective, identifying the correct blockmodel is crucial and both methods agree on the column clusters. Each algorithm uncovered one small cluster (cluster 1) of three organizations that were heavy signers of advertisements in four of the five categories, with the EC ads being the exception. Even so, one of these three organizations, the Earth Island Institute, signed in all five categories (22 separate advertisements in total), including all of the EC ads. The other dominant organizational signer was the International Forum on Globalization, which also signed in all five areas. Both algorithms also extracted a small cluster (cluster 2) of four organizations that were heavy signers of ads in each of the GE, EG, and EC categories, as well as a cluster (cluster 3) of 10 organizations that were heavy signers in both the GE and IA categories. The TMKLMH and TMGA algorithms also produced identical clusters of organizations that were heavy signers in only one category (GE, cluster 4), (EG, cluster 5), and EC (cluster 8). This was to be expected as some of the organizations had core interests that led them to sign in a single area. This was detected using both methods.

The disparity between the TMKLMH and TMGA solution occurred in clusters 6, 7, and 9. Cluster 6 corresponded to organizations that were heavy signers of only the IA ads. Whereas the TMGA solution contained 13 organizations in cluster 6, the TMKLMH solution contained only 10. Contrastingly, the TMKLMH solution placed 18 organizations in cluster 7, which corresponded to heavy signers of only the TM ads, whereas the TMGA solution had only 10 organizations in this cluster. The reported ARI value is driven by the differences in these three clusters of organizations. Again from a blockmodeling perspective, differences in row cluster memberships are problematic with regard to interpreting the stances of organizations. However, the cores of these clusters show considerable consistency with the differences being due to those organizations signing much fewer advertisements.

5. Summary and extensions

This paper has two primary purposes. First, it brings to light an important, yet understudied, extension of minimum sum-of-squares clustering. Potential applications of the TMKLMP include (but are not limited to) the analysis of gene expression data, the formation of manufacturing cells, the investigation of buyer/supplier relationships in a supply chain, the study of vocabulary used in a set of documents, the participation of organizations in projects or political activities, and the social behavior of individuals in social networks. Much of the blockmodeling partitioning has been applied to binary networks. The methods introduced here can extend blockmodeling to tackle valued networks. Another approach in this area is provided by Žiberna (2007), particularly with homogeneity blockmodeling. Second, a new real-coded genetic algorithm for the TMKLMP has been introduced and shown to perform well relative to a multistart implementation of two-mode *KL*-means clustering (i.e., TMKLMH) that been purported as the best available heuristic procedure for the problem (van Rosmalen et al., 2009). Although our findings reinforce those of van Rosmalen et al. (2009) in the sense that TMKLMH generally performed well, the genetic algorithm was able to improve the TMKLMH result in about one-fifth of the test problems. Moreover, an implementation of the genetic algorithm that was comparable to TMKLMH with respect to computation time provides statistically superior results in terms of *VAF*. An examination of the evolution of *VAF* across different levels of restarts and genetic search iterations further exemplified the efficacy of the genetic algorithm.

Extensions to this research can be divided into three areas. The first area centers on the development of better methods for the TMKLMP. This might include the design of other heuristic approaches and initialization procedures. For example, it is possible that slightly modified initialization procedures could lead to better performance, particularly for the difficult 10% density problem instances (see Brusco et al., 2013a). There is also the potential for progress in the area of exact approaches (see Brusco and Doreian, in press). The second area focuses on data preparation. In applications for real-valued networks (e.g., journal co-citation data), it might be advisable to transform the raw matrix elements prior to implementing TMKLMP; however, the precise nature of such transformations remains an open research question. The third area for future research involves comparisons of different objective functions for heterogeneity blockmodeling. Although the sum-of-squares criterion has a good pedigree in the statistical literature, it is not necessarily the best approach in all circumstances.

References

Baier, D., Gaul, W., Schader, M., 1997. Two-mode overlapping clustering with applications in simultaneous benefit segmentation and market structuring. In: Klar, R., Opitz, O. (Eds.), Classification and Knowledge Organization. Springer, Heidelberg, pp. 557–566.

Borgatti, S.P., Everett, M.G., 1992. Regular blockmodels of multiway, multimode matrices. Social Networks 14, 91–120.

Borgatti, S.P., Everett, M.G., 1997. Network analysis of two-mode data. Social Networks 19, 243–269.

Both, M., Gaul, W., 1985. PENCLUS: Penalty clustering for marketing applications. Discussion Paper No. 82. Institution of Decision Theory and Operations Research, University of Karlsruhe, Karlsruhe.

Both, M., Gaul, W., 1987. Ein vergleich zweimodaler clusteranalyseverfahren. Methods Oper. Res. 57, 593–605.

Brusco, M., 2011. Analysis of two-mode network data using nonnegative matrix factorization. Social Networks 33, 201–210.

Brusco, M.J., Doreian, P., 2015. An exact algorithm for the two-mode KL-means partitioning problem. J. Classif. (in press).

Brusco, M., Doreian, P., Lloyd, P., Steinley, D., 2013a. A variable neighborhood search method for a two-mode blockmodeling problem in social network analysis. Network Sci. 1, 191–212.

Brusco, M., Doreian, P., Mrvar, A., Steinley, D., 2013b. An exact algorithm for blockmodeling of two-mode network data. J. Math. Sociol. 37, 61–84.

Brusco, M., Doreian, P., Steinley, D., Satornino, C.B., 2013c. Multiobjective blockmodeling for social network analysis. Psychometrika 78, 498–525.

Brusco, M.J., Steinley, D., 2007a. A comparison of heuristic procedures for minimum within-cluster sums of squares partitioning. Psychometrika 72, 583–600.

Brusco, M., Steinley, D., 2007b. A variable neighborhood search method for generalized blockmodeling of two-mode binary matrices. J. Math. Psychol. 51, 325–338.

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

Castillo, W., Trejos, J., 2002. Two-mode partitioning: Review of methods and application of tabu search. In: Jajuga, K., Sololowski, A., Bock, H. (Eds.), Classification, Clustering and Data Analysis. Springer, Berlin, pp. 43–51.

Clapham, C., 1996. The Concise Oxford Dictionary of Mathematics. Oxford University Press, New York, NY.

DeSarbo, W.S., 1982. GENNCLUS: new models for general nonhierarchical clustering analysis. Psychometrika 47, 449–475.

Doreian, P., Batagelj, V., Ferligoj, A., 2004. Generalized blockmodeling of two-mode network data. Soc. Networks 26, 29–53.

Doreian, P., Batagelj, V., Ferligoj, A., 2005. Generalized Blockmodeling. Cambridge University Press, Cambridge.

Doreian, P., Lloyd, P., Mrvar, A., 2013. Partitioning large signed two-mode networks: problems and prospects. Soc. Networks 35, 1–21.

Everett, M.G., Borgatti, S.P., 2013. The dual-projection approach for two-mode networks. Soc. Networks 35, 204–210.

Forgy, E.W., 1965. Cluster analyses of multivariate data: efficiency versus interpretability of classifications. Abstr. Biometrics 21, 768–769.

Gaul, W., Schader, M., 1996. A new algorithm for two-mode clustering. In: Bock, H., Polasek, W. (Eds.), Data Analysis and Information Systems. Springer, Berlin, pp. 15–23.

Hansohm, J., 2002. Two-mode clustering with genetic algorithms. In: Gaul, W., Ritter, G. (Eds.), Classification, Automation and New Media. Springer, Berlin, pp. 87–93.

Hartigan, J., 1972. Direct clustering of a data matrix. J. Am. Stat. Assoc. 67, 123–129.

Hubert, L., Arabie, P., 1985. Comparing partitions. J. Classif. 2, 193–218.

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

MacQueen, J.B., 1967. Some methods for classification and analysis of multivariate observations. In: Le Cam, L.M., Neyman, J. (Eds.), Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1. University of California Press, Berkeley, CA, pp. 281–297.

Madeira, S.C., Oliveira, A.L., 2004. Biclustering algorithms for biological data analysis: a survey. IEEE Trans. Comput. Biol. Bioinf. 1, 24–45.

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

Selim, H.M., Askin, R.G., Vakharia, A.J., 1998. Cell formation in group technology: review, evaluation and directions for future research. Comput. Ind. Eng. 34, 3–20.

Steinhaus, H., 1956. Sur la division des corps matériels en parties. Bulletin de l'Académie Polonaise des Sciences, 801–804, Classe III, IV(12).

Steinley, D., 2004. Properties of the Hubert–Arabie adjusted Rand index. Psychol. Methods 9, 386–396.

Trejos, J., Castillo, W., 2000. Simulated annealing optimization for two-mode partitioning. In: Gaul, W., Decker, R. (Eds.), Classification and Information at the Turn of the Millennium. Springer, Heidelberg, pp. 135–142.

van Mechelen, I., Bock, H.H., DeBoeck, P., 2004. Two-mode clustering methods: a structured overview. Stat. Methods Med. Res. 13, 363–394.

van Rosmalen, J., Groenen, P.J.F., Trejos, J., Castillo, W., 2009. Optimization strategies for two-mode partitioning. J. Classif. 26, 155–181.

van Uitert, M., Meuleman, W., Wessels, L., 2008. Biclustering sparse binary genomic data. J. Comput. Biol. 15, 1329–1345.

Vichi, M., 2001. Double *K*-means clustering for simultaneous classification of objects and variables. In: Borra, S., Rocchi, R., Schader, M. (Eds.), Advances in Classification and Data Analysis—Studies in Classification, Data Analysis and Knowledge Organization. Springer, Heidelberg, pp. 43–52.

Žiberna, A., 2007. Generalized blockmodeling of valued networks. Soc. Networks 29, 105–126.

Žiberna, A., 2009. Evaluation of direct and indirect blockmodeling of regular equivalence in valued networks by simulations. Metodološki Zvezki 6, 99–134.