Journal of Classification 32:481-515 (2015)
DOI: 10.1007/s00357-015-9185-z

# An Exact Algorithm for the Two-Mode *KL*-Means Partitioning Problem

Michael J. Brusco

Florida State University, Florida, USA

Patrick Doreian

University of Ljubljana, Ljubljana, Slovenia
University of Pittsburgh, Pennsylvania, USA

**Abstract:** Two-mode partitioning applications are increasingly common in the physical and social sciences with a variety of models and methods spanning these applications. Two-mode *KL*-means partitioning (TMKLMP) is one type of two-mode partitioning model with a conceptual appeal that stems largely from the fact that it is a generalization of the ubiquitous (one-mode) *K*-means clustering problem. A number of heuristic methods have been proposed for TMKLMP, ranging from a two-mode version of the *K*-means heuristic to metaheuristic approaches based on simulated annealing, genetic algorithms, variable neighborhood search, fuzzy steps, and tabu search. We present an exact algorithm for TMKLMP based on branch-and-bound programming and demonstrate its utility for the clustering of brand switching, manufacturing cell formation, and journal citation data. Although the proposed branch-and-bound algorithm does not obviate the need for approximation methods for large two-mode data sets, it does provide a first step in the development of methods that afford a guarantee of globally-optimal solutions for TMKLMP.

**Keywords**: Two-mode partitioning; Exact algorithms; Branch-and-bound.

---

Authors' Addresses: Michael J. Brusco, College of Business, Florida State University, Tallahassee, Florida, USA, email: mbrusco@fsu.edu; Patrick Doreian**,** Faculty of Social Sciences, University of Ljubljana, Ljubljana, Slovenia, and Department of Sociology, University of Pittsburgh, Pittsburgh, Pennsylvania, USA, email: pitpat@pitt.edu.

Published online: 26 September 2015

## 1. Introduction

Two-mode partitioning problems require establishing partitions for two distinct sets of objects. These problems have considerable relevance for a diverse range of disciplines. For example, in an effort to form manufacturing cells, an industrial engineer might seek to establish one partition for a set of manufactured parts and another partition for the set of machines that produce those parts (Selim, Askin, and Vakharia 1998). Similarly, in the context of analyzing gene expression microarray data, a biological scientist could use two-mode partitioning to simultaneously produce partitions for both genes and the cell tissues from which they were measured (Madeira and Oliveira 2004; Prelić et al. 2006: Van Uitert, Meuleman, and Wessels 2008). A psychological scientist might employ two-mode clustering to extract dominant interaction patterns from profile matrices (Schepers and van Mechelen 2011). The social network literature is replete with examples of two-mode partitioning, with applications including the participation of CEO's in clubs (Brusco, Doreian, Mrvar, and Steinley 2013), the attendance of women at social events (Doreian, Batagelj, and Ferligoj 2004), and the participation of civic organizations in community projects (Brusco and Steinley 2007a; Mische and Pattison 2000). Madeira and Oliveira (2004) provide an excellent review of two-mode partitioning methods from the biological sciences literature. Similarly, van Mechelen, Bock, and De Boeck (2004) provide a rigorous review of literature from the medical and psychological literature, whereas Doreian, Batagelj, and Ferligoj (2005) cover two-mode partitioning from the perspective of generalized blockmodeling of social networks.

Although the objective criteria and constraints for two-mode partitioning vary widely across different application domains, there is general agreement that they are typically formidable discrete optimization problems. Madeira and Oliveira (2004) observed that almost all interesting forms of two-mode clustering are NP-complete. Moreover, as noted by Van Rosmalen, Groenen, Trejos, and Castillo (2009, p. 159), the number of partitions in most two-mode clustering applications is enormous, "…and a complete enumeration of the possible solutions is almost always computationally infeasible." For these reasons, there has been a reliance on heuristic methods for two-mode partitioning problems.

One particular two-mode partitioning problem that has received considerable attention with respect to the development of heuristic procedures is a two-mode generalization of the classic *K*-means partitioning problem (Forgy 1965; MacQueen 1967; Steinhaus 1956; Steinley 2006). We refer to this problem as two-mode *KL*-means partitioning (TMKLMP). The term "*KL*-means" in this description is to reflect the fact that there are

*K* clusters assumed for the first set of objects and *L* clusters assumed for the second. The origins of TMKLMP date back (at least) to Hartigan's (1972) development of a divisive hierarchical algorithm for minimizing a variance-based measure in the context of two-mode clustering. Also noteworthy, is the pioneering work of DeSarbo (1982), Both and Gaul (1985, 1987), and Mirkin, Arabie, and Hubert (1995) with respect to the development of the additive biclustering model (see Wilderjans, Depril, and van Mechelen 2013). Heuristic approaches for TMKLMP include alternating exchanges (Gaul and Schader 1996), two-mode *K*-means (Baier, Gaul, and Schader 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 2007a), and fuzzy steps (Van Rosmalen et al. 2009). The most comprehensive comparison of heuristic methods to date was performed by Van Rosmalen et al. (2009) who found that a multistart implementation of the two-mode *K*-means heuristic often performed as well as, or better than, its more sophisticated competitors (simulated annealing, tabu search, fuzzy steps).

We acknowledge the statements expressed in past research concerning the size and complexity of two-mode clustering problems. Moreover, we support the continued development of heuristic methods because they will remain necessary for large-scale applications. Nevertheless, we perceive an opportunity for the introduction of *exact* methods for two-mode clustering that can provide guaranteed, globally-optimal solutions for problems of modest size.[1] There are several facets to the motivation for pursuing exact solution procedures. First, there is inherent mathematical value in a method that guarantees a global optimum as opposed to one that does not. Second, it is worthwhile for an applied analyst to be able to use the exact procedure for an application and report, without equivocation, that the solution is globally-optimal. Third, the exact algorithm can be used to produce benchmarks for small problems that can be used to measure the performance of approximation procedures. Fourth, it must be recognized that the capability for an exact solution of only modestly-sized problems today does not mean that only modestly-sized problems can be solved exactly tomorrow. Consider, for example, the recent advancements

[1] It is impossible to put precise limitations on the size of the problems that can be tackled because the computational feasibility depends on many factors, such as the number of row and column objects, the number of row and column clusters, the variation in the matrix elements, the software used to implement the algorithm, hardware platform, and how long the user is willing to wait for the optimal solution. A rough upper limit would be 40 or fewer total (row + column) objects, but this could be tightened or loosened depending on the other characteristics.

in mathematical programming approaches for one-mode *K*-means partitioning (Aloise, Hansen, and Liberti 2012). These have enabled the exact solution of problems with thousands of objects, which was unthinkable 15 years ago.

In this spirit, we present an exact algorithm for TMKLMP that uses branch-and-bound programming (Balas 1965; Land and Doig 1960). The algorithm is based on principles from Brusco's (2006) repetitive branch-and-bound algorithm for one-mode sum-of-squares (i.e., *K*-means) partitioning, which was recently extended to clusterwise regression by Carbonneau, Caporossi, and Hansen (2012).

The principal contribution of this paper is an exact algorithm for a specific two-mode partitioning problem. Although this contribution is methodological, not substantive, several examples are presented to demonstrate possible applications of the branch-and-bound algorithm for the TMKLMP. These application examples include brand switching (DeSarbo and De Soete 1984; Hoffman, van der Heijden, and Novak 2001; Rao and Sabavala 1981), part-machine clustering (Brusco and Steinley 2007b; Chan and Milner 1982; Selim, Askin, and Vakharia 1998), and journal citation analysis (Doreian 1985, 1988; Doreian and Fararo 1985). Although the brand switching and journal citation examples involve data that are inherently one-mode, it is often appropriate to treat the data as two-mode (see, for example, Doreian, Batagelj, and Ferligoj 2005, pp. 265–269). In the journal citation analysis application, the same set of journals defines the rows and the columns; however, the sending (cited) role of a journal need not be the same as its receiving (citing) role.

Section 2 provides a formal presentation of the TMKLMP. The branch-and-bound algorithm is described in Section 3. Computational results for the brand switching, part-machine clustering, and journal citation analysis applications are provided in Sections 4, 5, and 6, respectively. The paper concludes in Section 7 with a brief summary and a discussion of possible extensions.

## 2. Two-Mode *KL*-Means Partitioning

### 2.1 Notation

The notation used throughout the manuscript is divided into two subsections. In Section 2.1.1, we present the general notation for describing the two-mode *KL*-means partitioning problem. This notation comports closely with the one used in previous descriptions of the same problem (Brusco and Steinley 2007a; Van Rosmalen et al. 2009). In Section 2.1.2, the notation is specific to the branch-and-bound algorithm described in Section 3.

### 2.1.1 General Notation

$N$ := the number of row (mode 1) objects;
$M$ := the number of column (mode 2) objects;
$K$ := the number of row clusters;
$L$ := the number of column clusters;
$\Pi$ := the set of all $K$-cluster partitions of the row objects;
$\pi$ := a $K$-cluster partition of the row objects, $(\pi = \{S_1,\ldots,S_K\}) \in \Pi$, where $S_k$ is a set containing the row objects assigned to cluster $k$ and $N_k = |S_k|$ is the number of objects in $S_k$, for all $1 \le k \le K$;
$\Omega$ := the set of all $L$-cluster partitions of the column objects;
$\omega$ := an $L$-cluster partition of the column objects, $(\omega = \{T_1,\ldots,T_L\}) \in \Omega$, where $T_l$ is a set containing the column objects assigned to cluster $l$ and $M_l = |T_l|$ is the number of objects in $T_l$, for all $1 \le l \le L$;
$\mathbf{X}$ := a $\Re^{N\times M}$ two-mode proximity matrix;
$\bar{x}$ := the grand mean of $\mathbf{X}$, $\bar{x} = \dfrac{\sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij}}{NM}$;
$v$ := the total (sum-of-squares) variation in $\mathbf{X}$,

$$v = \sum_{i=1}^{N}\sum_{j=1}^{M}(x_{ij} - \bar{x})^2$$

$\bar{x}_{kl}$ := the average of the elements in the submatrix of $\mathbf{X}$ formed by the rows associated with the row objects in row cluster $S_k$ and the columns corresponding to the column objects in column cluster $T_l$,

$$\bar{x}_{kl} = \frac{\sum_{i\in S_k}\sum_{j\in T_l} x_{ij}}{N_k M_l}, \tag{1}$$

for all $1 \le k \le K$ and $1 \le l \le L$;
$v_{kl}$ := the within-submatrix sum-of-squared error variation associated with the row objects in row cluster $S_k$ and the column objects in column cluster $T_l$,

$$v_{kl} = \sum_{i\in S_k}\sum_{j\in T_l}(x_{ij} - \bar{x}_{kl})^2, \tag{2}$$

for all $1 \le k \le K$ and $1 \le l \le L$.

2.1.2 Specific Notation for the Branch-and-Bound Algorithm

$p$ := a position pointer indicating the object currently under consideration for assignment, which ranges from $1 \le p \le N + M$;
$n$ := a position pointer that marks row objects, $1 \le n \le N$;
$m$ := a position pointer that marks column objects, $1 \le m \le M$;
$q$ := the cluster for which object $p$ is currently under consideration for assignment;
$\delta$ := an $(N+M) \times 1$ vector that identifies each object in the ordered list of $N+M$ objects as either a row object with $\delta(p) = 1$, or a column object with $\delta(p) = 2$, for $1 \le p \le N + M$;
$\lambda_r$ := the number of empty row clusters;
$\lambda_c$ := the number of empty column clusters;
$f_{UB}$ := an upper bound on the sum-of-squares objective function;
$f$ := the objective function corresponding to the current partial solution in the branch-and-bound algorithm;
$\tau_{kl}$ := current sum of proximity elements in the submatrix associated with row cluster $k$ and column cluster $l$, for $1 \le k \le K$ and $1 \le l \le L$;
$\rho_{kl}$ := the total number of proximity elements in the submatrix associated with row cluster $k$ and column cluster $l$, for $1 \le k \le K$ and $1 \le l \le L$;

## 2.2 Formulation

The goal of TMKLMP is to produce partitions, $\pi$ and $\omega$, that minimize the total sum-of-squared error variation across all $KL$ submatrices. A formal statement of the TMKLMP optimization problem is:

$$\min_{\pi \in \Pi, \omega \in \Omega} : f(\pi, \omega) = \sum_{k=1}^{K} \sum_{l=1}^{L} v_{kl} \,. \tag{3}$$

A pair of partitions, $\pi$ and $\omega$, that minimizes $f(\pi, \omega)$, will also maximize the variation-accounted-for ($VAF$) in $\mathbf{X}$ across all submatrices. The $VAF$ is computed as follows:

$$VAF = vaf(\pi, \omega) = \frac{(v - f(\pi, \omega))}{v}. \tag{4}$$

The total number of possible solutions, $\xi(\Pi, \Omega)$, to a TMKLMP problem with $N$ row objects, $M$ column objects, $K$ row clusters, and $L$ column clusters is computed as follows:

$$\xi(\Pi,\Omega)=\left[\frac{1}{K!}\sum_{k=0}^{K}(-1)^k\binom{K}{k}(K-k)^N\right]\times\left[\frac{1}{L!}\sum_{l=0}^{L}(-1)^l\binom{L}{l}(L-l)^M\right] \quad (5)$$

The size of the solution space is defined by the product of two Stirling numbers of the second kind (Clapham 1996). The first term is the number of ways to partition $N$ row objects into $K$ row clusters, whereas the second term is the number ways to partition $M$ column objects into $L$ column clusters. Because any row partition can be matched with any column partition to produce a solution to TMKLMP, the total number of possible solutions is the product of these Stirling numbers. Table 1 provides the results of some solution space computations for various combinations of $N$, $M$, $K$, and $L$ with the simplifying restrictions $N=M$ and $K=L$. The table reveals that, even for what are considered very modestly-sized problems, the solution space for TMKLMP is enormous.

### 2.3 Two-Mode *KL*-Means Heuristic (TMKLMH)

Several authors have described adaptations of *K*-means clustering that can be used to solve the TMKLMP (Baier, Gaul, and Schader 1997; Brusco and Steinley 2007a; Van Rosmalen et al. 2009; Vichi 2001). The heuristic consists of the following steps:

Step 0. *Construct initial partitions*, π and ω, for the row and column object sets respectively, and compute $f^* = f(\pi,\omega)$.

Because the performance of the TMKLMH is sensitive to the initial partitions at Step 0, several authors have implemented the method using multiple (e.g., 500) restarts from different random initial partitions (Brusco and Steinley 2007a; Van Rosmalen et al. 2009) to mitigate the potential for a poor local minimum.

Step 1. *Reassignment of Row Objects*.

Step 1a: Compute $\bar{x}_{kl}$ for all $1 \le k \le K$ and $1 \le l \le L$.

Step 1b. 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 1c. 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 1d. 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.

Table 1. The number of possible solutions for various combinations of $N$, $M$, $K$, and $L$.

| $N$ | $M$ | $K$ | $L$ | Number of partitions for row objects (the same number for column objects because $M = N$ and $L = K$) | The approximate number of possible solutions, $\xi(\Pi,\Omega)$ |
|---|---|---|---|---|---|
| 10 | 10 | 3 | 3 | 9,330 | $8.705\times10^{07}$ |
| 10 | 10 | 4 | 4 | 34,105 | $1.163\times10^{09}$ |
| 12 | 12 | 3 | 3 | 86,526 | $7.487\times10^{09}$ |
| 12 | 12 | 4 | 4 | 611,501 | $3.739\times10^{11}$ |
| 14 | 14 | 3 | 3 | 788,970 | $6.225\times10^{11}$ |
| 14 | 14 | 4 | 4 | 10,391,745 | $1.080\times10^{14}$ |
| 14 | 14 | 5 | 5 | 40,075,035 | $1.606\times10^{15}$ |
| 16 | 16 | 3 | 3 | 7,141,686 | $5.100\times10^{13}$ |
| 16 | 16 | 4 | 4 | 171,798,901 | $2.951\times10^{16}$ |
| 16 | 16 | 5 | 5 | 1,096,190,550 | $1.202\times10^{18}$ |
| 19 | 19 | 3 | 3 | 193,448,101 | $3.742\times10^{16}$ |
| 19 | 19 | 4 | 4 | 11,259,666,950 | $1.268\times10^{20}$ |
| 19 | 19 | 5 | 5 | 147,589,274,710 | $2.178\times10^{22}$ |
| 19 | 19 | 6 | 6 | 693,081,601,779 | $4.804\times10^{24}$ |

The submatrix means are computed in Step 1a. The distance of each row object to the submatrix means for each row cluster are computed in Step 1b. Step 1c updates the row object assignments by assigning each row object to its nearest row cluster. There is the potential for empty row clusters after Step 1c. This is remedied by reassignment of the case that is farthest from its current cluster centroid to the empty cluster in Step 1d.

Step 2. *Reassignment of Column Objects*.

Step 2a: Compute $\bar{x}_{kl}$ for all $1 \le k \le K$ and $1 \le l \le L$.

Step 2b. 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 2c. 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 2d. 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.

The submatrix means are computed in Step 2a. The distance of each column object to the submatrix means for each column cluster are computed in Step 2b. Step 2c updates the column object assignments by assigning each column object to its nearest column cluster. The presence of empty column clusters after Step 2c can be remedied by reassignment of the case that is farthest from its current cluster centroid to the empty cluster in Step 2d.

Step 3. Compute $f(\pi,\omega)$. If $f(\pi,\omega) < f^*$, then set $f^* = f(\pi,\omega)$ and return to Step 1; otherwise, stop.

The objective function, $f(\pi,\omega)$, is computed in Step 3. If the sum-of-squares criterion has been reduced ($f(\pi,\omega) < f^*$), then control returns to Step 1; otherwise the algorithm terminates.

## 3. A Branch-and-Bound Algorithm

### 3.1 Background

Branch-and-bound programming is an exact solution procedure for discrete optimization problems that avoids exhaustive (or complete) enumeration of all possible solutions in the solution space. This is accomplished by implicitly eliminating a large number of solutions that cannot possibly lead to an optimal solution. To understand this process of implicit enumeration, it is necessary to distinguish between *complete* and *partial* solutions. As the names suggest, a *complete* solution affords a full or final solution to the optimization problem. For example, in our TMKLMP framework, a complete solution is one where all of the row and column objects have been placed in a cluster. By contrast, a *partial* solution occurs when some, but not all, of the objects have been assigned to clusters.

In a minimization context, the branch-and-bound process makes use of an incumbent (best-found) solution that provides an objective function value that is an *upper bound* for the global minimum value of the objective function. During the course of the algorithm, *lower bounds* are computed for partial (incomplete) solutions to the optimization problem, representing the minimum possible objective function value that could be achieved by completing the partial solution. If one of these lower bounds exceeds the upper bound, then that partial solution is pruned from the search tree and all possible complete solutions that could stem from that branch of the tree need not be explicitly evaluated. For any given application of the branch-and-bound method, among the key design features of the algorithm are the design of the search tree, the provision of a good initial upper bound, and the process of constructing good lower bounds for partial solutions.

The use of branch-and-bound programming methods for one-mode partitioning problems has an extensive history, spanning the fields of statistics, operations research, computer science and other disciplines (Brusco and Stahl 2005a, Chap. 3-6; Hansen and Delattre 1978; Klein and Aronson 1991; Koontz, Narendra, and Fukunaga 1975; Palubeckis 1997). Brusco et al. (2013) recently designed a two-mode partitioning algorithm for fitting structurally-equivalent blockmodels to social network data. In our design and presentation of a branch-and-bound procedure for TMKLMP, we draw heavily from this prior work, particularly that of Brusco et al. (2013) and Brusco and Stahl (2005a).

### 3.2 Steps of the Algorithm

For each step of the branch-and-bound algorithm, a precise mathematical statement of the step is provided along with a brief non-technical description. The steps of the branch-and-bound algorithm are as follows:

Step 0. *Upper Bound*. Choose $K$ and $L$ and obtain $f_{UB}$ using the TMKLMH in Section 2.3.

The branch-and-bound algorithm requires the provision of the number of row and column clusters ($K$ and $L$), as well as an initial upper bound on the optimal objective function value ($f_{UB}$). in Step 0. This process is external to the branch-and-bound algorithm itself. The initial upper bound is obtained in Step 0 using 500 restarts of the TMKLMH described in section 2.3.

Step 1. *Initialize Parameters*. Set $\lambda_r = K$, $\lambda_c = L$, $S = \varnothing$, $T = \varnothing$, $p = 0$, $n = 0$, $m = 0$, $\boldsymbol{\delta} = \mathbf{0}$, and $\tau_{kl} = 0$ and $\rho_{kl} = 0$ for $1 \le k \le K$ and $1 \le l \le L$. Go to Step 2.

An initialization of parameters occurs in Step 1. Initially, the number of empty clusters for rows and columns are $\lambda_r = K$ and $\lambda_c = L$, respectively, because no objects have yet been assigned. Of particular importance in the algorithm are the $\boldsymbol{\delta}$ parameters, which identify whether the object currently under consideration for assignment is a row (mode 1) object ($\delta(p) = 1$) or a column (mode 2) object ($\delta(p) = 2$). The algorithm alternates between assignment of row objects and column objects because this enables more rapid development of the bounds.[2] For example, if 10 row objects were assigned prior to any column objects, then the current partial

[2] In cases where $N > M$, the alternation occurs until all column objects are exhausted and, subsequently the remaining row objects are assigned. When $N < M$, the alternation occurs until all row objects are exhausted and, subsequently the remaining column objects are assigned.

solution would have a bound of zero because there are no column assignments to compute within-submatrix means and variances.

Step 2. *Advance Object Pointer*. Set $q = 1$ and $p = p + 1$. If $[(n \leq m \wedge n < N) \vee (m = M)]$, then go to Step 3; otherwise, go to Step 4.

The object pointer, $p$, is advanced in Step 2, and the cluster index is set to $q$ =1. A determination as to whether to add a row object in Step 3 or a column object in Step 4 is based on the (row or column) nature of the previously added object, as well as whether there are any row or column objects remaining. A row object is selected if either (a) the number of row objects assigned is currently equal to or less than the number of column objects assigned ($n \leq m$) and not all row objects have been assigned ($n < N$), or (b) all column objects have already been assigned ($m = M$). If neither condition (i.e., (a) or (b)) holds, then a column object is selected.

Step 3. *Row Object Assignment*.

Step 3a. Set $\delta(p) = 1$, $n = n + 1$, and $S_q = S_q \cup \{n\}$.

Step 3b. If $|S_q| = 1$, then set $\lambda_r = \lambda_r - 1$.

Step 3c. Update the $\tau_{kl}$ and $\rho_{kl}$ values as follows:

$$\tau_{ql} = \tau_{ql} + \sum_{j \in T_l} x_{nj} \quad \forall\ 1 \leq l \leq L,$$

$$\rho_{ql} = \rho_{ql} + |T_l| \quad \forall\ 1 \leq l \leq L,$$

Step 3d. If $N - n < \lambda_r$, then go to Step 7; otherwise go to Step 5.

Step 3a marks object $p$ as a row object by setting $\delta(p) = 1$, increments the index for the row object under consideration for assignment ($n = n + 1$), and assigns row object $n$ to row cluster $q$ by setting $S_q = S_q \cup \{n\}$. Step 3b reduces the number of empty row clusters by one ($\lambda_r = \lambda_r - 1$) if row object $n$ is the first object assigned to row cluster $q$ (i.e., $|S_q| = 1$). Step 3c dynamically updates the sum of the elements in the submatrices ($\tau_{kl}$), as well as the number of elements in the submatrices ($\rho_{kl}$). These updates only occur for column clusters, $l$, that are not empty (i.e., $T_l \neq \varnothing$). The dynamic updating of $\tau_{kl}$ and $\rho_{kl}$ each time a row object is added or removed (see Step 8b) is far less computationally demanding than recomputing these values from scratch each time it is necessary to compute bounds in Step 5. If $N - n < \lambda_r$ in Step 3d, then the partial solution is pruned (by passing control to Step 7) because there are not enough remaining row objects to fill up the empty row clusters.

Step 4. *Column Object Assignment*.

Step 4a. Set $\delta(p) = 2$, $m = m + 1$, and $T_q = T_q \cup \{m\}$.

Step 4b. If $|T_q| = 1$, then set $\lambda_c = \lambda_c - 1$..
Step 4c. Update the $\tau_{kl}$ and $\rho_{kl}$ values as follows:

$$\tau_{kq} = \tau_{kq} + \sum_{i \in S_k} x_{im} \quad \forall\ 1 \le k \le K,$$

$$\rho_{kq} = \rho_{kq} + |S_k| \quad \forall\ 1 \le k \le K.$$

Step 4d. If $M - m < \lambda_c$, then go to Step 7; otherwise go to Step 5.

Step 4 essentially completes the same processes for column objects that Step 3 performs for row objects. Step 4a marks object $p$ as a column object, increments the index for the column object under consideration for assignment, and assigns column object $m$ to column cluster $q$. Step 4b reduces the number of empty column clusters if column object $m$ is the first object assigned to column cluster $q$. Step 4c dynamically updates the $\tau_{kl}$ and $\rho_{kl}$ values for non-empty row clusters. If $M - m < \lambda_c$ in Step 4d, then the partial solution is pruned (by passing control to Step 7) because there are not enough remaining column objects to fill up the empty column clusters.

Step 5. *Pruning Test*.

Step 5a. Compute $\bar{x}_{kl}$ using Equation (1) for all $1 \le k \le K$ and $1 \le l \le L : S_k \ne \varnothing \wedge T_l \ne \varnothing$.
Step 5b. Compute $v_{kl}$ using Equation (2) for all $1 \le k \le K$ and $1 \le l \le L : S_k \ne \varnothing \wedge T_l \ne \varnothing$. Let $v_{kl} = 0$ for all $(k,l)$ pairs for which $S_k = \varnothing \vee T_l = \varnothing$.
Step 5c. Compute $f$ using Equation (3).
Step 5d. If $f \ge f_{UB}$, then go to Step 7; otherwise go to Step 6.

The submatrix means ($\bar{x}_{kl}$), submatrix sum-of-squared errors ($v_{kl}$), and objective function value ($f$) for the current partial solution are computed in Steps 5a, 5b, and 5c, respectively. It is important to note that $\bar{x}_{kl}$ and $v_{kl}$ are only computed for $(k,l)$ for which both row cluster $k$ and column cluster $l$ are nonempty. If $f \ge f_{UB}$ at Step 5d, then the current partial solution is pruned (by passing control to Step 7) because completion of the row and cluster assignments cannot possibly lead to a better objective function value than $f_{UB}$. If $f < f_{UB}$ at Step 5d, then control is passed to Step 6.

Step 6. *Update Best-Found Solution*? If $(n = N \wedge m = M)$, then set $f_{UB} = f$, $\pi = \{S_1,..., S_K\}$, $\omega = \{T_1, ..., T_L\}$, and go to Step 7; otherwise go to Step 2.

If $n = N$ and $m = M$ at Step 6, then the solution is complete (i.e., all row and column objects are assigned). The complete solution is stored as the best-found (incumbent) solution by setting $f_{UB} = f$, $\pi = \{S_1,..., S_K\}$, and $\omega = \{T_1, ..., T_L\}$. Control is then passed to Step 7 for dispensation. If the solution is not complete, then processing returns to Step 2 for assignment of the next object.

Step 7. *Dispensation*. If $[(\delta(p) = 1 \wedge q = K) \vee (\delta(p) = 2 \wedge q = L) \vee (\delta(p) = 1 \wedge |S_q| = 1 \wedge |S_{q+1}| = 0) \vee (\delta(p) = 2 \wedge |T_q| = 1 \wedge |T_{q+1}| = 0)]$, then proceed to Step 10; otherwise, proceed to Step 8.

The dispensation of a solution in Step 7 requires a determination as to whether a *branch-right* operation or *depth retraction* should occur. A branch-right operation involves the reassignment of the current object, $p$, to the next cluster[3]. This can only occur if there are clusters available on the right and if the move of object $p$ from cluster $q$ to cluster $q + 1$ will not make cluster $q$ empty. This latter condition is vital because it prevents the evaluation of clustering solutions that differ only with respect to cluster labeling (see Klein and Aronson 1991; Brusco and Stahl 2005a, p. 23). Thus, in total, there are four conditions that would prohibit a branch right operation: (a) no available row clusters on the right ($\delta(p) = 1$ and $q = K$), (b) no available column clusters on the right ($\delta(p) = 2$ and $q = L$), (c) the emptying of a row cluster (($\delta(p) = 1$ and $|S_q| = 1$), or (d) the emptying of a column cluster ($\delta(p) = 2$ and $|T_q| = 1$). If any of these four conditions holds, then control is passed to Step 10 for depth retraction.

Step 8. *Branch on Row Object*.

Step 8a. If $\delta(p) = 2$, then go to Step 9; otherwise, proceed with Step 8b.

Step 8b. Remove contribution of the assignment of row object $n$ to cluster $q$:

$$\tau_{ql} = \tau_{ql} - \sum_{j \in T_l} x_{nj} \quad \forall\ 1 \le l \le L,$$

$$\rho_{ql} = \rho_{ql} - |T_l| \quad \forall\ 1 \le l \le L,$$

Step 8c. Set $S_q = S_q - \{n\}$. If $|S_q| = 0$, then set $\lambda_r = \lambda_r + 1$.

Step 8d. Set $q = q + 1$, $S_q = S_q \cup \{n\}$. If $|S_q| = 1$, then set $\lambda_r = \lambda_r - 1$.

Step 8e. Add contribution of the assignment of row object $n$ to cluster $q$:

[3] The reassignment is from cluster $q$ to cluster $q$+1, that is, the cluster on the 'right', hence the term 'branch-right'.

$$\tau_{ql} = \tau_{ql} + \sum_{j \in T_l} x_{nj} \quad \forall\ 1 \le l \le L,$$

$$\rho_{ql} = \rho_{ql} + |T_l| \quad \forall\ 1 \le l \le L,$$

Step 8f. If $N - n < \lambda_r$, then go to Step 7; otherwise go to Step 5.

If $\delta(p) = 2$ at Step 8a, then control is passed to Step 9 to branch right on the column object. Step 8b removes the contributions of row object $n$ to $\tau_{ql}$ and $\rho_{ql}$ (for $1 \le l \le L$) based on its current assignment to row cluster $q$. Step 8c removes row object $n$ from row cluster $q$ and increments $\lambda_r$ if this creates an empty cluster. Step 8d increments the row cluster ($q = q + 1$) and assigns row object $n$ to that row cluster ($S_q = S_q \cup \{n\}$). If row object $n$ becomes the first object in row cluster $q$, then the number of empty clusters is reduced by one ($\lambda_r = \lambda_r - 1$). The contributions of row object $n$ to $\tau_{ql}$ and $\rho_{ql}$ (for $1 \le l \le L$) based on its new cluster assignment are performed in Step 8e. If $N - n < \lambda_r$ in Step 8f, then the partial solution is pruned (by passing control to Step 7) because there are not enough remaining row objects to fill up the empty row clusters; otherwise control returns to Step 5 to perform the pruning test on the new partial solution.

Step 9. *Branch on Column Object.*

Step 9a. Remove contribution of the assignment of column object $m$ to cluster $q$:

$$\tau_{kq} = \tau_{kq} - \sum_{i \in S_k} x_{im} \quad \forall\ 1 \le k \le K,$$

$$\rho_{kq} = \rho_{kq} - |S_k| \quad \forall\ 1 \le k \le K.$$

Step 9b. Set $T_q = T_q - \{m\}$. If $|T_q| = 0$, then set $\lambda_c = \lambda_c + 1$.

Step 9c. Set $q = q + 1$, $T_q = T_q \cup \{m\}$. If $|T_q| = 1$, then set $\lambda_c = \lambda_c - 1$.

Step 9d. Add contribution of the assignment of row object $m$ to cluster $q$:

$$\tau_{kq} = \tau_{kq} + \sum_{i \in S_k} x_{im} \quad \forall\ 1 \le k \le K,$$

$$\rho_{kq} = \rho_{kq} + |S_k| \quad \forall\ 1 \le k \le K.$$

Step 9e. If $M - m < \lambda_c$, then go to Step 7; otherwise go to Step 5.

Step 9 completes the same processes for column objects that Step 8 executes for row objects. Step 9a removes the contributions of column object $m$ to the $\tau_{kq}$ and $\rho_{kq}$ values. Step 9b removes column object $m$ from column cluster $q$ and increments $\lambda_c$ if this creates an empty cluster. Step 9c increments the column cluster, assigns column object $m$ to that column cluster,

and, if appropriate, reduces the number of empty clusters by one. The contributions of column object $m$ to the $\tau_{kq}$ and $\rho_{kq}$ values based on its new cluster assignment are performed in Step 9d. If $M - m < \lambda_c$ in Step 9e, then the partial solution is pruned; otherwise control returns to Step 5 to perform the pruning test on the new partial solution.

Step 10. *Depth Retraction on Row Object*.

Step 10a. If $\delta(p) = 2$, then go to Step 11; otherwise, proceed with Step 10b.

Step 10b. Remove contribution of the assignment of row object $n$ to cluster $q$:

$$\tau_{ql} = \tau_{ql} - \sum_{j \in T_l} x_{nj} \quad \forall\ 1 \le l \le L,$$

$$\rho_{ql} = \rho_{ql} - |T_l| \quad \forall\ 1 \le l \le L,$$

Step 10c. Set $S_q = S_q - \{n\}$. If $|S_q| = 0$, then set $\lambda_r = \lambda_r + 1$.

Step 10d. Set $p = p - 1$ and $n = n - 1$. If $p = 0$, then return $f_{\mathrm{UB}}$, $\pi$, $\omega$, and STOP; otherwise, proceed to Step 10e.

Step 10e. If $\delta(p) = 1$, then set $q = q' : n \in S_{q'}$; otherwise, set $q = q' : m \in T_{q'}$. Return to Step 7.

If $\delta(p) = 2$ at Step 10a, then control is passed to Step 11 for depth retraction based on a column object. Depth retraction[4] for a row object begins in Step 10b with the removal of the contributions of row object $n$ to $\tau_{ql}$ and $\rho_{ql}$ (for $1 \le l \le L$) based on its current assignment to row cluster $q$. Step 10c removes row object $n$ from row cluster $q$ and increments $\lambda_r$ if this creates an empty cluster. Step 10d reduces the object pointer index ($p = p - 1$) and the row objet index ($n = n - 1$). If depth retraction reduces the pointer index to $p = 0$ at Step 10d, then all of the possible solutions have been either implicitly or explicitly evaluated and the algorithm terminates with $\pi$ and $\omega$ as the optimal row and column partitions, respectively. If $p > 0$ at Step 10d, then Step 10e resets $q$ as the cluster index corresponding to object $p$ and passes control to Step 7.

Step 11. *Depth Retraction on Column Object*.

Step 11a. Remove contribution of the assignment of column object $m$ to cluster $q$:

[4] The term 'depth retraction' refers to the fact that the algorithm 'retracts' or 'backs up' in the search process by removing the assignment of the current object $p$ and returning to consideration of the previous object $p$-1.

$$\tau_{kq} = \tau_{kq} - \sum_{i \in S_k} x_{im} \quad \forall\ 1 \le k \le K,$$

$$\rho_{kq} = \rho_{kq} - |S_k| \quad \forall\ 1 \le k \le K.$$

Step 11b. Set $T_q = T_q - \{m\}$. If $|T_q| = 0$, then set $\lambda_c = \lambda_c + 1$.

Step 11c. Set $p = p - 1$ and $m = m - 1$. If $p = 0$, then return $f_{UB}$, $\pi$, $\omega$, and STOP; otherwise, proceed to Step 11d.

Step 11d. If $\delta(p) = 1$, then set $q = q' : n \in S_{q'}$; otherwise, set $q = q' : m \in T_{q'}$. Return to Step 7.

Step 11 essentially completes the same processes for column objects that Step 10 performs for row objects. Depth retraction for a column object begins in Step 11a with the removal of the contributions of column object $m$ to the $\tau_{kq}$ and $\rho_{kq}$ values. Step 11b removes column object $m$ from column cluster $q$ and increments $\lambda_c$ if this creates an empty cluster. Step 11c reduces the object pointer index and the column objet index. If depth retraction reduces the pointer index to $p = 0$ at Step 11c, then the algorithm terminates; otherwise Step 11d resets $q$ as the cluster index corresponding to object $p$ and passes control to Step 7.

### 3.3 Improved Bounding Procedures

The pruning test in Step 5 of the branch-and-bound algorithm can be enhanced using principles similar to those applied in one-mode $K$-means partitioning (Brusco 2006; Koontz, Narendra, and Fukunaga 1975). For any given partial assignment of $p = n + m$ objects, it is sometimes possible to substantially improve the lower bound computed in Step 5c by obtaining exact solutions for subcomponents of the objects yet to be assigned. Figure 1 helps to visualize this principle. Component 1 in this Figure consists of the $p = n + m$ objects that have been assigned to comprise the current partial solution. The computation of $f$ in Step 5c is for Component 1, and it represents a lower bound on the best-possible objective function value that can be achieved after assignment of the remaining ($N$+$M$-$p$) objects.

Now consider Component 2, which consists of matrix elements corresponding to the $n$ assigned row objects and the $M$-$m$ yet unassigned column objects. We do not know the final contribution of Component 2 to the objective function because the column objects have not been assigned. However, if we could efficiently obtain an optimal $K$-row, $L$-cluster two-mode clustering solution for the Component 2 submatrix, then this would be a lower-bound on the contribution that could stem from that component. Accordingly, we could add this lower bound for Component 2 to the lower

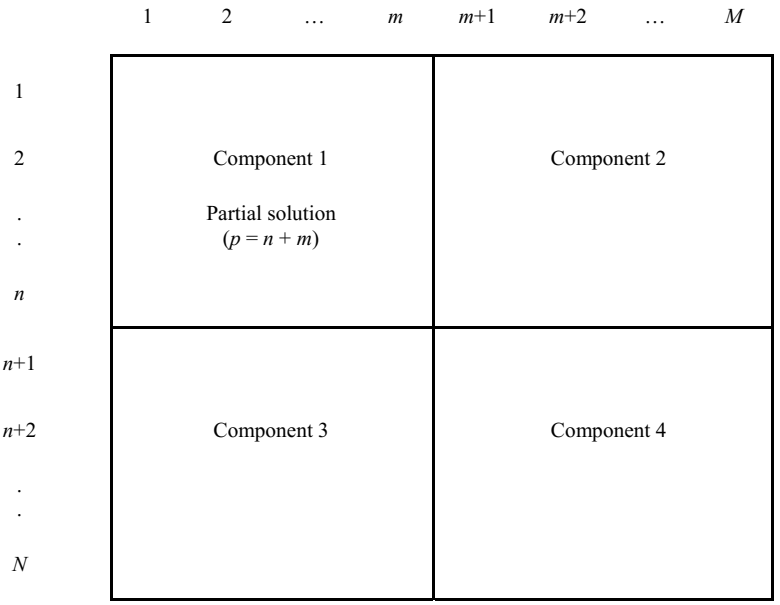


Figure 1. The four bound components of a partial solution.

bound for Component 1 when performing the pruning test in Step 5d of the algorithm. Using similar logic, lower bounds for Components 3 and 4 in Figure 1 could be established. Elements in Component 3 correspond to assigned column objects and yet unassigned row objects, whereas elements in Component 4 correspond to unassigned row objects and unassigned column objects. If optimal sum-of-squares partitions can be obtained from Components 2, 3, and 4 (with objective values $f_2$, $f_3$, and $f_4$, respectively), then the pruning test in Step 5d can be replaced by: $f+f_2+f_3+f_4 \geq f_{UB}$, which reflects a considerable strengthening of the lower bound.

As an example, consider an application where $K = L = 3$ and $N = M = 19$. Prior to running the branch-and-bound algorithm on the full matrix, we could use branch-and-bound to obtain exact solutions for all Component 4 submatrices of size $4 \times 4$ through $10 \times 10$. To be matched with the $4 \times 4$ Component 4 submatrix, exact solutions would also be obtained for the $15 \times 4$ Component 2 submatrix and the $4 \times 15$ Component 3 submatrix. Similarly, the bound for the $10 \times 10$ Component 4 submatrix would be augmented by the exact solutions for the $9 \times 10$ Component 2 submatrix and the $10 \times 9$ Component 3 submatrix.

### 3.4 Algorithm Implementation

The branch-and-bound algorithm has been written in Fortran 90 and is implemented on a laptop computer with a 2.13 GHz Intel P6200 processor with 4.0 GB of RAM. Different versions of the algorithm are available to include or ignore the elements on the main diagonal (when present) as desired. Because of the nature of some of our applications, it is important to accommodate the flexibility to ignore the main diagonal.[5] This flexibility is afforded by a version of the algorithm that ignores main diagonal terms when dynamically updating the submatrix sums ($\tau_{kl}$ and $\rho_{kl}$) in Steps 3c, 4c, 8b, 8e, 9a, 9d, 10b, and 11a.

### 3.5 Numerical Example

To illustrate the algorithm, we use a small synthetic numerical example with only $N = M = 3$ row and column objects. The cell values of the data matrix (**X**) are shown in the top panel of Figure 2. The main diagonal of **X** is ignored in the analysis. The branch-and-bound algorithm was applied to the data in the top panel of Figure 2 assuming $K = L = 2$ and an initial upper bound of $f_{UB} = \infty$ at Step 0. Table 2 displays the results of the branch-and-bound process for each partial and complete solution generated. The bottom panel of Figure 2 displays the optimal partitions of row and column objects.

The partial or complete solutions produced by the algorithm are indexed on the left-hand-side of Table 2. The cluster assignments for row objects (r1, r2, r3) and column objects (c1, c2, c3) for each solution are also provided. Notice that the rows and columns are interwoven in the sequence r1-c1-r2-c2-r3-c3. Because $K = L = 2$, the cluster assignments for the objects are displayed as either 0, 1, or 2, with 0 used to indicate the lack of an assignment for the object at the present status of the algorithm. The action taken on each partial solution is briefly described in the far right-hand column of Table 2.

The action taken on partial solutions 0 through 4 in Table 2 is to branch right. However, at partial solution 5, all three row objects are assigned to row cluster 1 and, therefore, the second row cluster cannot be

[5] The decision of whether or not to exclude the main diagonal (when present) is application dependent. For example, in journal citation analysis, the main diagonal elements (self-citations) are often large and can dominate the clustering solution if included. Exclusion of the main diagonal can be justified on the basis of emphasizing the flow of citations among journals. Similarly, in the context of brand switching, exclusion of the main diagonal is sensible if the focus is on the flow of switching among brands. However, if brand loyalty is also of interest, then perhaps the main diagonal (purchase of the same brand as last time) should be retained.

| object | 1 | 2 | 3 |
|---|---|---|---|
| 1 | --- | 12 | 4 |
| 2 | 7 | --- | 9 |
| 3 | 8 | 1 | --- |

| object | 1 | 3 | 2 |
|---|---|---|---|
| 1 | --- | 4 | 12 |
| 2 | 7 | 9 | --- |
| 3 | 8 | --- | 1 |

Figure 2. A Small Numerical Example for Illustrating the Algorithm – The Top Panel Contains the Raw Data and the Bottom Panel Displays the Optimal $K = L = 2$ Partition

filled. Therefore, pruning of this partial solution occurs at Step 3d of the algorithm. The first feasible complete solution occurs at solution 8, and this becomes the incumbent solution with $f_{UB} = 49.5$. This solution is later replaced with a better incumbent (solution 14) that has an objective function value of $f_{UB} = 12.667$. Solutions 21, 22, and 25 are noteworthy because they reveal occurrences of the pruning operation at Step 5d based on the fact that the objective function value for the current partial solution exceeds $f_{UB} = 12.667$. However, solution 27 is a complete solution with $f_{UB} = 2.0$ and is established as the new incumbent. This solution is ultimately confirmed as the global optimum when the algorithm terminates at solution 33. The optimal partition of row objects is {r1} and {r2, r3}, whereas the optimal partition of column objects is {c1, c3} and {c2}. The partition is visually displayed in the bottom panel of Figure 2. It is clear that the {r2, r3} row cluster and {c1, c3} column cluster yields a submatrix with the elements 7, 8, and 9, which average to 8. The within-submatrix sum-of-squares is, therefore, $(7-8)^2 + (8-8)^2 + (9-8)^2 = 2$, which is the value of $f_{UB}$. The other submatrices do not contribute anything to $f_{UB}$ because they only contain one element.

## 4 Application 1: Brand Switching

### 4.1 Application 1a: A Smaller Set of Brands

Our first application of the TMKLMP pertains to the context of brand switching in the field of marketing research. The raw data correspond to an asymmetric brand switching matrix, $\mathbf{G} = [g_{ij}]$ of consumption

Table 2. Partial and complete solutions evaluated for the numerical example.

| Solution Number | Assignments r1 | c1 | r2 | c2 | r3 | c3 | Processing of Partial/Complete Solution |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | branch forward |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | branch forward |
| 2 | 1 | 1 | 0 | 0 | 0 | 0 | branch forward |
| 3 | 1 | 1 | 1 | 0 | 0 | 0 | branch forward |
| 4 | 1 | 1 | 1 | 1 | 0 | 0 | branch forward |
| 5 | 1 | 1 | 1 | 1 | 1 | 0 | prune at Step 3d because cannot have empty row cluster, branch right on row object at Step 8 |
| 6 | 1 | 1 | 1 | 1 | 2 | 0 | branch forward |
| 7 | 1 | 1 | 1 | 1 | 2 | 1 | prune at Step 4d because cannot have empty column cluster, branch right on column object at Step 9 |
| 8 | 1 | 1 | 1 | 1 | 2 | 2 | new incumbent at Step 6, $f_{UB} = 49.5$, retract at Step 11 |
| 9 | 1 | 1 | 1 | 1 | 2 | 0 | retract at Step 10 |
| 10 | 1 | 1 | 1 | 1 | 0 | 0 | prune at Step7 and branch right on column object at Step 9 |
| 11 | 1 | 1 | 1 | 2 | 0 | 0 | branch forward |
| 12 | 1 | 1 | 1 | 2 | 1 | 0 | prune at Step 3d because cannot have empty row cluster, branch right on row object at Step 8 |
| 13 | 1 | 1 | 1 | 2 | 2 | 0 | branch forward |
| 14 | 1 | 1 | 1 | 2 | 2 | 1 | new incumbent at Step 6, $f_{UB} = 12.667$, branch right on column object at Step 9 |
| 15 | 1 | 1 | 1 | 2 | 2 | 2 | suboptimal, 32.667 > 12.667, retract at Step 11 |
| 16 | 1 | 1 | 1 | 2 | 2 | 0 | retract at Step 10 |
| 17 | 1 | 1 | 1 | 2 | 0 | 0 | retract at Step 11 |
| 18 | 1 | 1 | 1 | 0 | 0 | 0 | prune at Step 7 and branch right on row object at Step 8 |
| 19 | 1 | 1 | 2 | 0 | 0 | 0 | branch forward |
| 20 | 1 | 1 | 2 | 1 | 0 | 0 | branch forward |
| 21 | 1 | 1 | 2 | 1 | 1 | 0 | prune at Step 5d because 62 > 12.667, branch right on row object at Step 8 |
| 22 | 1 | 1 | 2 | 1 | 2 | 0 | prune at Step 5d because 28.667 > 12.667, retract at Step 10 |
| 23 | 1 | 1 | 2 | 1 | 0 | 0 | prune at Step 7 and branch right on column object at Step 9 |
| 24 | 1 | 1 | 2 | 2 | 0 | 0 | branch forward |
| 25 | 1 | 1 | 2 | 2 | 1 | 0 | prune at Step 5d because 60.5 > 12.667, branch right on row object as Step 8 |
| 26 | 1 | 1 | 2 | 2 | 2 | 0 | branch forward |
| 27 | 1 | 1 | 2 | 2 | 2 | 1 | new incumbent at Step 6, $f_{UB} = 2.0$, branch right on column object at Step 9 |
| 28 | 1 | 1 | 2 | 2 | 2 | 2 | suboptimal, 64.5 > 2.0, retract at Step 11 |
| 29 | 1 | 1 | 2 | 2 | 2 | 0 | retract at Step 10 |
| 30 | 1 | 1 | 2 | 2 | 0 | 0 | retract at Step 11 |
| 31 | 1 | 1 | 2 | 0 | 0 | 0 | retract at Step 10 |
| 32 | 1 | 1 | 0 | 0 | 0 | 0 | retract at Step 11 because c1 cannot be assigned to column cluster 2 |
| 33 | 1 | 0 | 0 | 0 | 0 | 0 | retract at Step 10 because r1 cannot be assigned to row cluster 2 and terminate at Step 10d |

Note – the partial or complete solutions are indexed at left. The "Assignments" correspond to cluster assignments for the row objects (r1, r2, r3) and column objects (c1, c2, c3), and an assignment of '0' indicates that the object is not yet assigned to a cluster. New incumbent solutions are identified at solution numbers 8, 14, and 27, with the latter reflecting the globally-optimal solution.

of $N = M = 8$ soft drinks at the two time periods (t1 and t2) measured (see Bass, Pessemier, and Lehmann 1972; DeSarbo and De Soete 1984, p. 602). The rows correspond to t1 and the columns to t2. Following the process used by DeSarbo and De Soete (1984), the brand switching matrix was normalized to obtain the input data matrix, **X**, by dividing each element in **G** by the product of the arithmetic means for the corresponding row and column. The branch-and-bound (main diagonal ignored) algorithm was applied to matrix **X** for all combinations of $2 \leq K \leq 3$ and $2 \leq L \leq 3$. Given the modest size of the network, the improved bounding strategy described in section 3.4 was not employed for this application. The optimal solutions were obtained in less than one second in all instances. The partitions for the $K = L = 2$ and $K = L = 3$ implementations are displayed in the top and bottom panels, respectively, of Figure 3.

The optimal $K = L = 2$ solution in the top panel of Table 2 consists of identical partitions of soft drinks at periods t1 and t2. For both periods, there is a cluster of non-diet drinks plus Fresca (i.e., Coke, Pepsi, 7-Up, Sprite, Fresca) and the remaining diet drinks (Tab, Like, Diet Pepsi). This corresponds to one of the two-cluster partitions obtained by DeSarbo and De Soete (1984, Figure B) using hierarchical clustering. The $K = L = 2$ solution has a *VAF* of 49.75%. Increasing only the number of clusters for t1 (or t2) results in modest improvements in the *VAF*. For example, the optimal ($K = 2$, $L = 3$) solution yields *VAF* = 57.38%, whereas the optimal ($K = 3$, $L = 2$) solution provides *VAF* = 56.66%. By contrast, increasing the number of clusters to 3 for both t1 and t2 (i.e., $K = L = 3$) improves the *VAF* to 76.58%. The optimal $K = L = 3$ solution in the bottom panel of Figure 3 reveals that this substantial improvement is achieved by splitting the diet drink clusters for t1 and t2 in the optimal $K = L = 2$ solution (note that the non-diet cluster is unchanged). The two largest elements in **X** correspond to switches from Tab at t1 to Like at t2 (25.3187), and Diet Pepsi at t1 to Tab at t2 (22.1264). In the optimal $K = L = 3$ partition for t1, Diet Pepsi is peeled off as a singleton cluster, whereas Like is a singleton cluster in the partition for t2. This enables the isolation of the two largest elements in **X** as the only element in their respective submatrices, thus greatly improving the *VAF*.

### 4.2 Application 1b: A Larger Set of Brands

To assess the scalability of the branch-and-bound algorithm for a larger set of brands, we considered a second empirical brand switching dataset pertaining to ownership of $N = M = 15$ automobiles in the United Kingdom (Colombo, Ehrenberg, and Sabavala 1994; Hoffman, van der Heijden, and Novak 2001). In this instance, the raw data were available in the form of a frequency matrix, $\mathbf{C} = [c_{ij}]$, pertaining to the previous (rows)

| | Coke | Pepsi | 7-UP | Sprite | Fresca | Tab | Like | Diet Pepsi |
|---|---|---|---|---|---|---|---|---|
| Coke | xxx | 5.6126 | 5.3333 | 4.1855 | 3.2451 | 1.1896 | 2.3209 | 1.3550 |
| Pepsi | 7.1924 | xxx | 6.5794 | 5.7075 | 3.3352 | 0.9517 | 2.1099 | 2.7101 |
| 7-UP | 7.5581 | 5.8639 | xxx | 7.5339 | 4.1465 | 0.5948 | 4.5011 | 1.2508 |
| Sprite | 4.6324 | 6.5759 | 9.2212 | xxx | 7.7521 | 3.4498 | 4.9934 | 3.0228 |
| Fresca | 9.1835 | 6.1571 | 4.6355 | 8.1427 | Xxx | 6.3048 | 7.5253 | 6.9837 |
| Tab | 3.2508 | 3.3508 | 5.9813 | 3.0440 | 7.2113 | xxx | 25.3187 | 8.3388 |
| Like | 3.5352 | 10.0105 | 7.5763 | 3.2723 | 9.8254 | 10.3494 | xxx | 13.6547 |
| Diet Pepsi | 3.7790 | 4.8586 | 2.3427 | 7.0773 | 10.4563 | 22.1264 | 6.5407 | xxx |

| | Coke | Pepsi | 7-UP | Sprite | Fresca | Tab | Diet Pepsi | Like |
|---|---|---|---|---|---|---|---|---|
| Coke | xxx | 5.6126 | 5.3333 | 4.1855 | 3.2451 | 1.1896 | 1.3550 | 2.3209 |
| Pepsi | 7.1924 | xxx | 6.5794 | 5.7075 | 3.3352 | 0.9517 | 2.7101 | 2.1099 |
| 7-UP | 7.5581 | 5.8639 | xxx | 7.5339 | 4.1465 | 0.5948 | 1.2508 | 4.5011 |
| Sprite | 4.6324 | 6.5759 | 9.2212 | xxx | 7.7521 | 3.4498 | 3.0228 | 4.9934 |
| Fresca | 9.1835 | 6.1571 | 4.6355 | 8.1427 | xxx | 6.3048 | 6.9837 | 7.5253 |
| Tab | 3.2508 | 3.3508 | 5.9813 | 3.0440 | 7.2113 | xxx | 8.3388 | 25.3187 |
| Like | 3.5352 | 10.0105 | 7.5763 | 3.2723 | 9.8254 | 10.3494 | 13.6547 | xxx |
| Diet Pepsi | 3.7790 | 4.8586 | 2.3427 | 7.0773 | 10.4563 | 22.1264 | xxx | 6.5407 |

Figure 3. The $K = L = 2$ (Top Panel) and $K = L = 3$ (Bottom Panel) Partitions for the Brand Switching Example from DeSarbo and De Soete (1984).

and current (columns) automobile brand ownership for a sample of 18,140 individuals. We used a transformation process proposed by Rao and Sabavala (1981) to obtain the proximity matrix (**X**) from the raw switching frequencies (**C**) as follows:

$$x_{ij} = \frac{c_{ij} c_{..}}{c_{i.} c_{.j}}, \text{ for all } 1 \le i \le N \text{ and } 1 \le j \le M; \tag{6}$$

where $c_{..}$ is the total number of individuals in the sample, $c_{i.}$ is the sum of the elements in row $i$ ($1 \le i \le N$) and $c_{.j}$ is the sum of elements in column $j$ ($1 \le j \le M$).

The branch-and-bound algorithm (main diagonal ignored) was applied to **X** for all combinations of $K$ and $L$ on the intervals $3 \leq K \leq 5$ and $3 \leq L \leq 5$. The computation times and *VAF* values for each combination are shown in Table 3. The algorithm was extremely efficient, requiring a maximum of only 38.03 seconds for the ($K = L = 5$) solution. Parameter settings of $K = L = 3$ produced a solution with *VAF* = 73.21%. Only slight improvement to *VAF* = 77.23% was realized when increasing $L$ to 4 (i.e., $K = 3$, $L = 4$); however, a greater improvement to *VAF* = 86.20% was obtained from increasing $K$ to 4 (i.e., $K = 4$, $L = 3$). Increases of $K$ and $L$ beyond the ($K = 4$, $L = 3$) setting yielded fairly modest improvement in *VAF*. For example, the ($K = 5$, $L = 3$) setting yielded *VAF* = 87.86%, and the ($K = L = 4$) setting produced *VAF* = 89.13%. The optimal solution for the ($K = 4$, $L = 3$) setting is displayed in Figure 4. The dashed vertical line in Figure 4 makes it possible to display the solution for the ($K = L = 4$) setting in the same figure.

The optimal ($K = 4$, $L = 3$) partition in Figure 4 reveals some very small clusters. For example, there are three singleton clusters of previously-owned automobiles: {Honda}, {Mercedes}, and {BMW}, and all other brands are in the remaining cluster. With respect to currently-owned automobile clusters, {Honda, Mercedes} form one small cluster, {Toyota} is a singleton cluster, and all other brands are in the remaining cluster. If a fourth cluster for currently-owned vehicles is permitted, then {BMW, Saab} are extracted from the large cluster. The automobile brands that comprise the small clusters in Figure 4 (i.e., Honda, Mercedes, BWM, Toyota, BMW, and Saab) are those that often occupy extreme positions in the mapping approaches employed by Hoffman, van der Heijden, and Novak (2001).

Toyota is extracted as a singleton current-ownership cluster because of the high degree of switching that occurs from Honda to Toyota, which also helps to explain why Honda is a singleton previous-ownership cluster. The Mercedes brand also forms a singleton previous-ownership cluster because of its high-degree of switching to Honda. The BMW brand is a singleton previous-ownership cluster because of its high-degree of switching to Honda and Mercedes, which also helps explain why {Honda, Mercedes} forms a small current-ownership cluster. When the number of current-ownership clusters is permitted to increase to $L = 4$, the {BMW, Saab} cluster emerges largely because of the high-degree of switching from BMW to Saab, as well as from Mercedes to both BMW and Saab.

## 5. Application 2: Part-Machine Clustering

A well-studied problem in the field of operations management and industrial engineering pertains to formation of manufacturing cells. In this

Table 3. Computational results for the larger brand switching matrix

| No. of Clusters | VAF | CPU Time |
|---|---|---|
| $K = 3, L = 3$ | .7321 | 3.87 |
| $K = 3, L = 4$ | .7723 | 6.35 |
| $K = 3, L = 5$ | .7842 | 7.32 |
| $K = 4, L = 3$ | .8620 | 6.70 |
| $K = 4, L = 4$ | .8913 | 10.13 |
| $K = 4, L = 5$ | .9028 | 10.21 |
| $K = 5, L = 3$ | .8786 | 7.29 |
| $K = 5, L = 4$ | .9138 | 11.66 |
| $K = 5, L = 5$ | .9253 | 38.03 |

| | Honda | Merced | Toyota | BMW | Saab | Citro | Fiat | Ford | GM | Nissan | Peug | Renlt | Rover | VW | Volvo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Honda | xxx | 1.1161 | 7.4687 | .9560 | .0000 | 1.2637 | .6925 | .2122 | .1948 | 1.6554 | .4838 | 1.1806 | .4501 | .4061 | 1.0733 |
| Mercedes | 13.1859 | xxx | .0000 | 1.7527 | 1.9015 | .2896 | .0000 | .1946 | .0714 | .3571 | .1109 | .8116 | .1375 | 1.1167 | .7028 |
| BMW | 4.7385 | 5.1474 | .0000 | xxx | 2.6573 | .8094 | .2662 | .3955 | .1597 | .1996 | .1240 | .2269 | .3267 | .9363 | .6286 |
| Toyota | .5137 | .2790 | xxx | .4780 | .0000 | .4739 | .2597 | .2701 | .2143 | 2.0449 | .6047 | .5903 | .3751 | .3553 | .4600 |
| Saab | 1.2843 | 1.3952 | 2.2406 | .7967 | xxx | 1.5796 | .8657 | .5065 | .2435 | .0000 | .9071 | .7378 | .3751 | .6345 | 1.5333 |
| Citroen | .6165 | .6697 | .0000 | .3824 | 1.2446 | xxx | 1.4543 | .3241 | .3740 | .9933 | 1.4513 | 1.1510 | .4051 | .3045 | .1840 |
| Fiat | .2580 | .1402 | 1.4255 | .3201 | .2605 | 1.1108 | xxx | .4264 | .3718 | .5625 | .9416 | .6671 | .5935 | .4589 | .3466 |
| Ford | .1233 | .2344 | .3675 | .4494 | .2178 | .3697 | .3117 | xxx | .4629 | .4821 | .4391 | .3365 | .3849 | .3716 | .3865 |
| GM | .1302 | .3183 | .4732 | .3836 | .3286 | .4404 | .2633 | .3924 | xxx | .2530 | .5288 | .4114 | .4278 | .4310 | .4372 |
| Nissan | .1726 | .0938 | .7530 | .2677 | .1743 | .6370 | .4364 | .4441 | .3469 | xxx | .9551 | .6447 | .3844 | .4094 | .6956 |
| Peugeot | .1884 | .2046 | .0000 | .3505 | .3803 | 1.5638 | .6031 | .5200 | .6357 | .7141 | xxx | .9469 | .8252 | .5211 | .8714 |
| Renault | .2193 | .2382 | .6377 | .2721 | .6642 | 1.2138 | 1.0348 | .3624 | .2827 | .8106 | 1.0068 | xxx | .5124 | .8234 | .5564 |
| Rover | .3257 | .2780 | .2841 | .4184 | .2818 | .6294 | .8310 | .5530 | .6103 | .5864 | .7612 | .4945 | xxx | .4597 | .4652 |
| VW | .4731 | 1.0279 | .2751 | 1.1250 | 1.4327 | .6304 | .6112 | .3376 | .3947 | .7025 | .8911 | .5436 | .2879 | xxx | .5413 |
| Volvo | .5632 | 1.2237 | 1.0645 | .8734 | 1.9899 | .6061 | .1424 | .2644 | .2349 | .0000 | .6298 | .7280 | .4524 | .5843 | xxx |

Figure 4. The ($K = 4$, $L = 3$) and $K = L = 4$ Partitions for the Larger Brand Switching Example from Hoffman, Van der Heijden, and Novak (2001). The dashed vertical line reveals the fourth cluster for the $K = L = 4$ solution.

particular application area, the two modes are parts and machines. The elements of the data matrix are $x_{ij} = 1$ if part $j$ requires processing on machine $i$ and 0 otherwise, for $1 \leq i \leq N$ and $1 \leq j \leq M$. The goal is to partition the parts and machines into $K$ and $L$ clusters, respectively, whereby families of machines producing similar parts are established. To illustrate, we consider an example from Chan and Milner (1982, Fig. 3a), which consists of $N = 10$ machines that produce $M = 15$ parts. The raw data are displayed in the top panel of Figure 5.

The branch-and-bound algorithm was applied to these data under the assumption of $K = L = 3$ clusters, which has been previously used for these data in the literature (Brusco and Steinley 2007b; Chan and Milner 1982). The optimal partition was obtained in 64 seconds and the resulting objective function value is 9.367, which yields $VAF = 71.61\%$. The partitions of parts and machines is displayed in convenient fashion in the bottom panel of Figure 4. These are the same partitions of parts and machines obtained by Brusco and Steinley (2007b) using a permutation procedure. Along the main diagonal are three submatrices of mostly ones, whereas the submatrices off of the main diagonal contain mostly zeros. In the manufacturing cell formation literature, the zeros in the main diagonal submatrices are called 'voids' and indicate parts that do not require processing on the machines that are members of their cell (or family). By contrast, the ones in the submatrices off the main diagonal are called 'exceptional elements' and indicate parts that require processing on machines that are not members of their cell. Ideally, voids and exceptional elements should be minimized, but their relative importance can vary. In this particular example, the branch-and-bound algorithm minimizing within-submatrix sum-of-squares happened to produce a solution that also minimizes the sum of the voids and exceptional elements (there are 6 voids and 5 exceptional elements in the solution). This was verified using a method developed by Brusco et al. (2013).

## 6. Application 3: Journal Citation Data

The elements of a journal citation frequency matrix, $\mathbf{C} = [c_{ij}]$, represent the number of times that journal $j$ cited journal $i$ within some prespecified time frame.[6] The analysis of journal citation networks can be accomplished using a variety of methods, such as unidimensional scaling (Brusco and Stahl 2005b), multidimensional scaling (Groenen and Heiser

[6] In the networks we used for our applications, the rows are the 'cited' journals (the senders) and the columns are the 'citing' journals (the receivers); however, this convention is reversed for many applications in the literature to reflect the idea of 'producer' and 'consumer' journals.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 3 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 7 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 8 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 9 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

| | 1 | 3 | 4 | 6 | 14 | 2 | 7 | 11 | 12 | 5 | 8 | 9 | 10 | 13 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

Figure 5. The Part-Machine Incidence Matrix from Chan and Milner (1982, Fig 3a) in the Top Panel and the $K = L = 3$ Partition for the Data (Bottom Panel).

1996), matrix factorization (Boyd et al. 2010; Brusco 2011), and cluster analysis (Doreian 1985, 1988; Doreian and Fararo 1985). We consider a journal citation matrix extracted for illustrative purposes from a larger matrix originally published by Groenen and Heiser (1996). The matrix consists of citations among $N = M = 19$ journals that publish quantitative social sciences research: *Psychometrika* (Pmetrika), *British Journal of Mathematical and Statistical Psychology* (BJMSP), *Multivariate Behavioral*

*Research* (MBR), *Applied Psychological Measurement* (APM), *Educational and Psychological Measurement* (EPM), *Journal of Educational Statistics* (JES), *Journal of Classification* (JClass), *Sociological Methodology* (SM), *Mathematical Social Sciences* (MSS), *Journal of Mathematical Psychology* (JMP), *Journal of Educational Measurement* (JEM), *Annual Review of Psychology* (ARS), *Psychological Review* (PRev), *Psychological Bulletin* (PB), *Psychological Reports* (PRep), *Perceptual and Motor Skills* (PMS), *Perception and Psychophysics* (PP), *Personality and Individual Differences* (PID), and *Experimental Aging Research* (EAR).

We explored the application of TMKLMP to the raw citation matrix. These problems were often easy to solve because a few really large entries in a network dominate the sum-of-squares measure. Nevertheless, the resulting solutions are not especially compelling substantively. As was the case for the brand switching applications in Section 4, a normalization process is absolutely necessary. Some authors have used logarithmic transformations of the matrix elements (Boyd et al. 2010; Brusco 2011); however, this is rather arbitrary and difficult to justify theoretically. Normalizing the matrix elements by row sums, column sums, or via an iterative double normalization process (Doreian 1985, 1988; Mosteller 1968) would seem to afford a more reasonable alternative. In our application, we applied the Rao and Sabavala (1981) transformation used in Section 4 (see Equation 6) to obtain **X** from the raw citation frequencies.

The branch-and-bound algorithm (main-diagonal ignored) was applied to **X** for all combinations of the number of clusters on the interval $3 \leq K \leq 5$ and $3 \leq L \leq 5$. In addition, based on this original set of runs, we also obtained a solution for the setting of $K = 6$ and $L = 6$. The computation times and *VAF* values for the nine combinations are reported in Table 4. The optimal row and column partitions for the $K = L = 4$, $K = L = 5$, and $K = L = 6$ solutions were selected for interpretation and are displayed in Table 5.[7] The detailed partition for the most complex model ($K = L = 6$) is displayed in Figure 6 for illustration purposes.

The results in Table 4 indicate that computation time typically increases as a function of the number of clusters. The computation time for the $K = L = 3$ setting was a modest 32.08 seconds. At $K = L = 4$, the computation time was markedly greater at 231.72 seconds. For $K = L = 5$, the computation time increased to nearly one hour (3473.84 seconds), and nearly three hours (9898.24 seconds) were required to solve the problem at $K = L = 6$. This finding that computation times rapidly increase as a function of the number of clusters is concordant with those reported for

[7] Our focus in this paper is primarily on the computational demands associated with different number of clusters. We refer readers to the work of Schepers, Ceulemans, and van Mechelen (2008) for a formal treatment of model selection in two-mode clustering.

Table 4. Computational results for the journal citation data

| No. of Clusters | VAF | CPU Time |
|---|---|---|
| $K = 3, L = 3$ | .4523 | 32.08 |
| $K = 3, L = 4$ | .4918 | 151.27 |
| $K = 3, L = 5$ | .5269 | 399.34 |
| $K = 4, L = 3$ | .4878 | 214.95 |
| $K = 4, L = 4$ | .5822 | 231.72 |
| $K = 4, L = 5$ | .6187 | 1131.46 |
| $K = 5, L = 3$ | .5071 | 1763.68 |
| $K = 5, L = 4$ | .6185 | 1559.12 |
| $K = 5, L = 5$ | .6675 | 3473.84 |
| $K = 6, L = 6$ | .7432 | 9898.24 |

implementation of similar branch-and-bound approaches for other clustering problems (Brusco 2006; Carbonneau, Caporossi, and Hansen 2012).

The findings in Table 4 also show that simultaneous increases in $K$ and $L$ seem to be necessary for obtaining significant improvements in *VAF*. For example, the $K = L = 3$ setting yielded *VAF* = 45.23%. Increasing only $K$ to 4 (i.e., $K = 4$, $L = 3$) improved *VAF* to only 49.18%. Similarly, increasing only $L$ to 4 (i.e., $K = 3$, $L = 4$) improved *VAF* to only 48.78%. However, a simultaneous increase of $K$ and $L$ to the $K = L = 4$ setting produced *VAF* = 58.22%. A similar *VAF* improvement pattern occurs when considering the move from $K = L = 4$ to $K = L = 5$. When moving from $K = L = 5$ to $K = L = 6$, the *VAF* improves from 66.75% to 74.32%.

The two mathematical social sciences journals (JMP and MSS) are isolated as both sending and receiving clusters in both the $K = L = 4$ to $K = L = 5$ solutions, and are also isolated as a cluster in the partition of receiving journals in the $K = L = 6$ solution. These two journals arise as singleton clusters in the partition of sending journals in the $K = L = 6$ solution. All three solutions in Table 5 also contain ARP as a singleton receiving journal cluster and {JClass, EAR} as a sending journal cluster. The reason for this result is that the vast majority of the citations for JClass and EAR were in ARP and, accordingly, the transformation process of Rao and Sabavala (1981) produces large values in **X** for the JClass row and ARP column, as well as the EAR row and ARP column. The $K = L = 4$ solution contains a solid core of statistically-oriented sending journals: {BJMSP, SM, MBR,

Table 5. Optimal solutions for the journal citation data

| | $K = L = 4$ | $K = L = 5$ | $K = L = 6$ |
|---|---|---|---|
| Sending (Cited) Journal Partitions | JMP | JMP | JMP |
| | MSS | MSS | MSS |
| | Jclass | Jclass | Jclass |
| | EAR | EAR | EAR |
| | BJMSP | BJMSP | BJMSP |
| | SM | SM | SM |
| | MBR | MBR | MBR |
| | JEM | JEM | PB |
| | APM | APM | Pmetrika |
| | JES | JES | JEM |
| | Pmetrika | Pmetrika | APM |
| | ARP | ARP | JES |
| | EPM | EPM | ARP |
| | PRev | PRev | EPM |
| | PB | PB | PRev |
| | PP | PP | PP |
| | PRep | PRep | PRep |
| | PID | PID | PID |
| | PMS | PMS | PMS |
| Receiving (Citing) Journal Partitions | JMP | JMP | JMP |
| | MSS | MSS | MSS |
| | ARP | ARP | ARP |
| | EAR | EAR | EAR |
| | BJMSP | BJMSP | BJMSP |
| | SM | SM | SM |
| | MBR | MBR | MBR |
| | JEM | JEM | JEM |
| | APM | APM | APM |
| | JES | JES | JES |
| | Pmetrika | Pmetrika | Pmetrika |
| | EPM | EPM | EPM |
| | JClass | JClass | JClass |
| | PRev | PRev | PRev |
| | PB | PB | PB |
| | PP | PP | PP |
| | Prep | PRep | PRep |
| | PID | PID | PID |
| | PMS | PMS | PMS |

Note – The hashed lines separate the clusters of the sending (top panel) and receiving (bottom panel) journals.

| | MSS | JMP | ARP | BJMSP | SM | EAR | MBR | JEM | APM | JES | Pmetrika | EPM | Jclass | Prev | PB | pp | Prep | PID | PMS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSS | xxx | 6.466 | 1.823 | 0.383 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 1.556 | 0.378 | 0.151 | 0.000 | 0.000 | 0.000 | 0.000 |
| JMP | 13.583 | xxx | 1.007 | 0.668 | 0.000 | 0.000 | 0.316 | 0.000 | 0.779 | 0.206 | 1.687 | 0.052 | 1.899 | 1.716 | 0.263 | 0.441 | 0.000 | 0.021 | 0.103 |
| Jclass | 1.023 | 0.000 | 9.859 | 0.239 | 1.269 | 0.000 | 0.452 | 0.000 | 0.000 | 0.000 | 2.641 | 0.000 | xxx | 0.000 | 0.094 | 0.000 | 0.000 | 0.000 | 0.000 |
| EAR | 0.000 | 0.000 | 4.960 | 0.000 | 0.000 | xxx | 0.492 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 1.231 | 0.287 | 0.415 | 0.481 | 0.480 |
| BJMSP | 1.673 | 0.564 | 0.689 | xxx | 4.610 | 2.509 | 2.707 | 0.960 | 1.842 | 3.751 | 1.577 | 1.423 | 0.353 | 0.172 | 0.752 | 0.191 | 0.000 | 0.320 | 0.027 |
| SM | 0.000 | 0.000 | 0.000 | 8.144 | xxx | 3.734 | 2.197 | 0.000 | 0.247 | 2.153 | 0.734 | 0.726 | 0.000 | 0.000 | 0.733 | 0.000 | 0.093 | 0.072 | 0.000 |
| Pmetrika | 0.945 | 0.482 | 0.630 | 3.198 | 1.547 | 1.474 | 2.436 | 1.561 | 3.016 | 2.779 | xxx | 1.102 | 4.304 | 0.174 | 0.320 | 0.102 | 0.288 | 0.320 | 0.206 |
| MBR | 0.000 | 0.000 | 0.307 | 2.552 | 4.309 | 3.475 | xxx | 0.000 | 1.394 | 0.477 | 0.732 | 1.448 | 0.000 | 0.076 | 0.548 | 0.000 | 0.646 | 1.308 | 0.238 |
| PB | 0.066 | 0.499 | 1.061 | 0.923 | 1.919 | 2.222 | 1.671 | 0.453 | 0.701 | 1.851 | 0.158 | 1.512 | 0.187 | 1.109 | xxx | 0.301 | 0.894 | 0.719 | 0.657 |
| JEM | 0.000 | 0.000 | 0.080 | 0.707 | 0.000 | 0.000 | 0.191 | xxx | 5.034 | 4.985 | 1.594 | 2.600 | 0.410 | 0.100 | 0.239 | 0.000 | 0.482 | 0.093 | 0.186 |
| APM | 0.330 | 0.000 | 0.122 | 2.005 | 0.818 | 1.485 | 1.456 | 7.240 | xxx | 1.665 | 1.947 | 2.947 | 0.000 | 0.000 | 0.607 | 0.000 | 0.245 | 0.379 | 0.024 |
| JES | 0.000 | 0.000 | 0.000 | 4.917 | 1.305 | 1.578 | 0.464 | 6.337 | 3.128 | xxx | 3.492 | 1.151 | 0.000 | 0.000 | 0.387 | 0.000 | 0.000 | 0.000 | 0.151 |
| ARP | 0.000 | 1.673 | xxx | 0.276 | 0.244 | 0.296 | 0.261 | 0.339 | 0.098 | 0.284 | 0.654 | 0.575 | 0.374 | 2.455 | 1.884 | 0.634 | 0.769 | 0.481 | 0.424 |
| EPM | 0.000 | 0.064 | 0.311 | 0.638 | 0.781 | 1.103 | 1.715 | 1.807 | 1.301 | 0.908 | 0.232 | xxx | 0.399 | 0.000 | 0.386 | 0.000 | 1.601 | 0.845 | 0.558 |
| Prev | 1.148 | 2.546 | 2.451 | 0.130 | 0.196 | 0.475 | 0.140 | 0.136 | 0.255 | 0.000 | 0.409 | 0.130 | 0.376 | xxx | 1.683 | 1.380 | 0.545 | 0.341 | 0.671 |
| PP | 0.443 | 0.723 | 1.060 | 0.064 | 0.000 | 0.000 | 0.000 | 0.000 | 0.051 | 0.049 | 0.201 | 0.000 | 0.065 | 0.864 | 0.457 | xxx | 0.013 | 0.059 | 0.538 |
| Prep | 0.000 | 0.000 | 0.155 | 0.000 | 0.000 | 0.157 | 0.184 | 0.045 | 0.155 | 0.075 | 0.000 | 0.552 | 0.000 | 0.048 | 0.797 | 0.000 | xxx | 1.208 | 1.042 |
| PID | 0.000 | 0.000 | 0.207 | 0.000 | 0.000 | 0.000 | 0.173 | 0.000 | 0.000 | 0.000 | 0.000 | 0.183 | 0.000 | 0.000 | 0.206 | 0.000 | 0.759 | xxx | 0.169 |
| PMS | 0.000 | 0.080 | 0.016 | 0.000 | 0.000 | 0.198 | 0.019 | 0.000 | 0.000 | 0.000 | 0.016 | 0.128 | 0.000 | 0.344 | 0.775 | 0.424 | 0.678 | 0.505 | xxx |

Note – The cell values were obtained using the Rao and Sabavala (1981) transformation to the raw citation frequencies.

Figure 6. The $K = 6$, $L = 6$ Solution for the Journal Citation Data

JEM, APM, JES, Pmetrika}. This cluster is split in the $K = L = 5$ solution as {BJMSP, SM, MBR} and {JEM, APM, JES, Pmetrika}. Arguably, this split is not conceptually intuitive because Pmetrika is perhaps more similar to BJMSP and MBR as a sending journal. Perhaps the best sending-journal clusters of the statistically-oriented journals arise in the $K = L = 6$ solution: {BJMSP, SM, MBR, Pmetrika, PB} and {JEM, APM, JES}. The first of these clusters consists of statistically-oriented psychology journals[8], whereas the second is comprised of statistically-oriented education journals. Five of the general psychology journals {PRev, PP, PRep, PID, PMS} form the core of a cluster for both sending and receiving journals in all three solutions in Table 5. These five journals are augmented with PB in all but the sending-journal partition for the $K = L = 6$ solution. The five journals are also augmented with ARP in all three sending-journal partitions, and with JClass in all three receiving-journal partitions.

## 7. Summary and Extensions

This paper presents an initial attempt to develop an exact solution procedure for TMKLMP. Specifically, we developed a branch-and-bound algorithm for TMKLMP, which can also be extended to apply exact solutions to submatrices of the dataset in an effort to improve the quality of the bounds. Moreover, a modified version of the algorithm was developed for TMKLMP applications to inherently one-mode datasets, which are common in network analysis applications. In these applications, it is often appropriate to ignore the main diagonal of the network, and the branch-and-bound algorithm is easily adapted for this flexibility.

We report results for TMKLMP applications to two brand switching datasets from the marketing research literature. For the smaller of these two datasets, which consisted of $N = M = 8$ soft drink brands, the branch-and-bound algorithm produced globally-optimal solutions in less than one second for all combinations of $K$ and $L$ evaluated. For the larger dataset that consisted of $N = M = 15$ brands, the computational requirements were slightly greater, but the maximum computation time (for the $K = L = 5$ setting) was less than one minute. The required computation time for the part-machine clustering application was also approximately one minute. The final application pertained to an $N = M = 19$ journal citation matrices. The computation times for this larger matrix were more substantial, yet still very reasonable at all combinations of $K$ and $L$. For this larger network, the

[8] Although *Psychological Bulletin* (PB) has not published statistically-oriented papers since the American Psychological Association introduced *Psychological Methods* in 1996, PB was a major outlet for statistically-oriented papers at the time (1991-1992) these citation data were compiled.

$K = L = 3$ problem was solved in less than one minute of computation time. Increasing the number of clusters to $K = L = 4$ increased the computation time to roughly four minutes. The computational requirements for the $K = L = 5$ and $K = L = 6$ settings were roughly one and three hours, respectively.

Once again, we emphasize that the proposed method does not supplant the efficient and effective heuristic procedures that have been developed for this problem (Baier, Gaul, and Schader 1996; Brusco and Steinley 2007a; Van Rosmalen et al. 2009). These heuristic methods are far more efficient and will likely produce globally-optimal partitions for problems of the size studied in this paper. Nevertheless, we hope that our contribution will spark further development of exact methods for TMKLMP. For example, extension of the mathematical programming approaches for one-mode *K*-means partitioning (see Aloise, Hansen, and Liberti 2012) would seem to offer a valuable avenue for future research.

Another extension of the research presented herein is the further development of exact two-mode partitioning approaches for applications related to brand switching, part-machine clustering, journal citation analysis, and other substantive problems. As noted previously, our primary focus in this paper centered on the design of an exact algorithm for a specific two-mode sum-of-squares partitioning problem. However, the best criterion for any given application is not necessarily the within-submatrix sum-of-squares. Although the basic structure of the branch-and-bound algorithm described in Section 3 is extensible to a variety of alternative objective criteria, such as inconsistency counts in the blockmodeling of social networks (Brusco et al. 2013) or part-machine grouping indices in cellular manufacturing (Selim, Askin, and Vakharia 1998), the ability to establish effective bounds can vary markedly across these criteria. Accordingly, our method can serve as a guideline for the general structure of algorithms for other two-mode partitioning applications, but substantial customization of the bound-construction process is apt to be necessary for such applications.

Finally, another fruitful avenue for future research pertains to the development and testing of normalization procedures. Regardless of whether an exact or heuristic algorithm is applied, the normalization process for applications such as brand switching and journal citation analysis can have a profound impact on the solution obtained, and this issue also warrants further study.

## References

ALOISE, D., HANSEN, P., and LIBERTI, L. (2012), "An Improved Column Generation Algorithm for Minimum Sum-of-Squares Clustering," *Mathematical Programming A*, *131*, 195–220.

BAIER, D., GAUL, W., and SCHADER, M. (1997), "Two-Mode Overlapping Clustering with Applications in Simultaneous Benefit Segmentation and Market Structuring," in *Classification and Knowledge Organization*, eds. R. Kar and O. Opitz, Heidelberg: Springer, pp. 557–566.

BALAS, E. (1965), "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," *Operations Research*, *13*, 517–546.

BASS, F.M., PESSEMIER, E.A., and LEHMANN, D.R. (1972), "An Experimental Study of Relationships Between Attitudes of Brand Preference and Choice," *Behavioral Science*, *17*, 532–541.

BOTH, M., and GAUL, W. (1985), "PENCLUS: Penalty Clustering for Marketing Applications," Discussion Paper No. 82, Institution of Decision Theory and Operations Research, University of Karlsruhe.

BOTH, M., and GAUL, W. (1987), "Ein Vergleich Zweimodaler Clusteranalyseverfahren," *Methods of Operations Research*, *57*, 593–605.

BOYD, J.P., FITZGERALD, W.J., MAHUTGA, M.C., and SMITH, D.A. (2010), "Computing Continuous Core/Periphery Structures for Social Relations Data with MINRES/SVD," *Social Networks*, 32, 125–137.

BRUSCO, M.J. (2006), "A Repetitive Branch-and-Bound Algorithm for Minimum Within-Cluster Sums of Squares Partitioning," *Psychometrika*, *71*, 347–363.

BRUSCO, M. (2011), "Analysis of Two-Mode Network Data Using Nonnegative Matrix Factorization," *Social Networks*, *33*, 201–210.

BRUSCO, M., DOREIAN, P., MRVAR, A., and STEINLEY, D. (2013), "An Exact Algorithm for Blockmodeling of Two-Mode Network Data," *Journal of Mathematical Sociology*, *37*, 61–84.

BRUSCO, M. ., and STAHL, S. (2005a), *Branch-and-Bound Applications in Combinatorial Data Analysis*, New York: Springer.

BRUSCO, M.J., and STAHL, S. (2005b), "Optimal Least-Squares Unidimensional Scaling: Improved Branch-and-Bound Procedures and Comparison to Dynamic Programming," *Psychometrika*, *70*, 253–270.

BRUSCO, M., and STEINLEY, D. (2007a), "A Variable Neighborhood Search Method for Generalized Blockmodeling of Two-Mode Binary Matrices," *Journal of Mathematical Psychology*, *51*, 325–338.

BRUSCO, M.J., and STEINLEY, D. (2007b), "Exact and Approximate Algorithms for Part-Machine Clustering Based on a Relationship Between Interval Graphs and Robinson Matrices," *IIE Transactions*, *39*, 925–935.

CARBONNEAU, R.A., CAPOROSSI, G., and HANSEN, P. (2012), "Extensions to the Repetitive Branch-and-Bound Algorithm for Globally Optimal Clusterwise Regression," *Computers and Operations Research*, *39*, 2748–2762.

CASTILLO, W., and TREJOS, J. (2002), "Two-Mode Partitioning: Review of Methods and Application of Tabu Search," in *Classification, Clustering and Data Analysis*, eds. K. Jajuga, A. Sololowski, and H. Bock, Berlin: Springer, pp. 43–51.

CHAN, H.M., and MILNER, D.A. (1982), "Direct Clustering Algorithm for Group Formation in Cellular Manufacturing," *Journal of Manufacturing Systems*, *1*, 65–74.

CLAPHAM, C. (1996), *The Concise Oxford Dictionary of Mathematics*, New York: Oxford University Press.

COLOMBO, R.A., EHRENBERG, A.S.C., and SABAVALA, D.J. (1994), "The Car Challenge: Diversity in Analyzing Brand Switching Tables," Working Paper, New York University.

DESARBO, W.S. (1982), "GENNCLUS: New Models for General Nonhierarchical Clustering Analysis," *Psychometrika*, *47*, 449–475.

DESARBO, W.S., and DE SOETE, G. (1984), "On the Use of Hierarchical Clustering for the Analysis of Nonsymmetric Proximities," *Journal of Consumer Research*, *11*, 601–610.

DOREIAN, P. (1985), "Structural Equivalence in a Psychology Journal Network," *Journal of the American Society for Information Science*, *36*, 411–417.

DOREIAN, P. (1988), "Testing Structural Equivalence Hypotheses in a Network of Geographical Journals," *Journal of the American Society for Information Science*, *39*, 79–85.

DOREIAN, P., BATAGELJ, V., and FERLIGOJ, A. (2004), "Generalized Blockmodeling of Two-Mode Network Data," *Social Networks*, *26*, 29–53.

DOREIAN, P., BATAGELJ, V., and FERLIGOJ, A. (2005), *Generalized Blockmodeling*, Cambridge: Cambridge University Press.

DOREIAN, P., and FARARO, T.J. (1985), "Structural Equivalence in a Journal Network," *Journal of the American Society for Information Science*, *36*, 28–37.

DOREIAN, P., LLOYD, P., and MRVAR, A. (2013), "Partitioning Large Signed Two-Mode Networks: Problems and Prospects," *Social Networks*, *35*, 178–203.

FORGY, E.W. (1965), "Cluster Analyses of Multivariate Data: Efficiency versus Interpretability of Classifications," Abstract in *Biometrics*, *21*, 768–769.

GAUL, W., and SCHADER, M. (1996), "A New Algorithm for Two-Mode Clustering," in H. Bock & W. Polasek (Eds.), *Data Analysis and Information Systems,* eds. H. Bock and W. Polasek, Berlin: Springer, pp. 15–23.

GROENEN, P.J.F., and HEISER, W.J. (1996), "The Tunneling Method for Global Optimization in Multidimensional Scaling," *Psychometrika*, *61*, 529–550.

HANSEN, P., and DELATTRE, M. (1978), "Complete-Link Cluster Analysis by Graph Coloring," *Journal of the American Statistical Association*, *73*, 397–403.

HANSOHM, J. (2002), "Two-Mode Clustering with Genetic Algorithms," in *Classification, Automation and New Media,* eds. W. Gaul and G. Ritter, Berlin: Springer, pp. 87–93.

HARTIGAN, J. (1972), "Direct Clustering of a Data Matrix," *Journal of the American Statistical Association*, *67*, 123–129.

HOFFMAN, D.L., VAN DER HEIJDEN, P.G.M., and NOVAK, T.P. (2001), "Mapping Asymmetry in Categorical Consumer Choice Data," Working Paper, Retrieved from http://www.academia.edu/2611216/Mapping_asymmetry_in_categorical_consumer_choice_data

HUBERT, L., and ARABIE, P. (1985), "Comparing Partitions," *Journal of Classification*, *2*, 193–218.

KLEIN, G., and ARONSON, J.E. (1991), "Optimal Clustering: A Model and Method," *Naval Research Logistics*, *38*, 447–461.

KOONTZ, W.L.G., NARENDRA, P.M., and FUKUNAGA, K. (1975), "A Branch and Bound Clustering Algorithm," *IEEE Transactions on Computing*, *C*-24, 908–915.

LAND, A.H., and DOIG, A. (1960), "An Automatic Method of Solving Discrete Programming Problems," *Econometrica*, *28*, 497–520.

MACQUEEN, J.B. (1967), "Some Methods for Classification and Analysis of Multivariate Observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 1), eds. L.M. Le Cam and J. Newman, Berkeley, CA: University of California Press, pp. 281–297.

MADEIRA, S.C., and 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., and HUBERT, L.J. (1995), Additive Two-Mode Clustering: The Error-Variance Approach Revisited," *Journal of Classification*, *12*, 243–263.

MISCHE, A., and PATTISON, P. (2000), "Composing a Civic Arena: Publics, Projects, and Social Settings," *Poetics*, *27*, 163–194.

MOSTELLER, F. (1968), "Association and Estimation in Contingency Tables," *Journal of the American Statistical Association*, *63*, 1–28.

PALUBECKIS, G. (1997), "A Branch-and-Bound Approach Using Polyhedral Results for a Clustering Problem," *INFORMS Journal on Computing*, *9*, 30–42.

PRELIĆ, A., BLUELER, S., ZIMMERMANN, P., WILLE, A., BÜHLMANN, P., GRUISSEM, W., HENNIG, L., THIELE, L., and ZITZLER, E. (2006), "A Systematic Comparison and Evaluation of Biclustering Methods for Gene Expression Data," *Bioinformatics*, *22*, 1122–1129.

RAO, V. R., and SABAVALA, D. J. (1981), "Inferences of Hierarchical Choice Processes from Panel Data," *Journal of Consumer Research*, *8*, 85–96.

RAO, V.R., SABAVALA, D.J., and LANGFELD, P.A. (1977), "Alternate Measures for Partitioning Analysis Based on Brand Switching Data," Working Paper, Cornell University.

SCHEPERS, J., CEULEMANS, E., and VAN MECHELEN, I. (2008), "Selection Among Multi-Mode Partitioning Models of Different Complexities," *Journal of Classification*, *25*, 67–85.

SCHEPERS, J., and 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., and VAKHARIA, A.J. (1998), "Cell Formation in Group Technology: Review, Evaluation and Directions for Future Research," *Computers and Industrial Engineering*, *34*, 3–20.

STEINHAUS, H. (1956), "Sur la Division des Corps Matériels en Parties," *Bulletin de l'Académie Polonaise des Sciences*, *Classe III, IV(12)*, 801–804.

STEINLEY, D. (2006), "*K*-means Clustering: A Half-Century Synthesis," *British Journal of Mathematical and Statistical Psychology*, *59*, 1–34.

TREJOS, J., and CASTILLO, W. (2000), "Simulated Annealing Optimization for Two-Mode Partitioning, in *Classification and Information at the Turn of the Millennium*, eds. W. Gaul and R. Decker, Heidelberg: Springer,. pp. 135–142.

VAN MECHELEN, I., BOCK, H.H., and 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., and CASTILLO, W. (2009), "Optimization Strategies for Two-Mode Partitioning," *Journal of Classification*, *26*, 155–181.

VAN UITERT, M., MEULEMAN, W., and WESSELS, L. (2008), "Biclustering Sparse Binary Genomic Data," *Journal of Computational Biology*, *15*, 1329–1345.

VICHI, M. (2001), "Double *K*-means Clustering for Simultaneous Classification of Objects and Variables," in *Advances in Classification and Data Analysis – Studies in Classification, Data Analysis and Knowledge Organization*, eds. S. Borra, R. Rocchi, and M. Schader, Heidelberg: Springer, pp. 43–52.

WILDERJANS, T.F., DEPRIL, D., and VAN MECHELEN, I. (2013), "Additive Biclustering: A Comparison of One New and Two Existing ALS Algorithms," *Journal of Classification*, *30*, 56–74.