Journal of Machine Learning Research 0 (0) 0-0 Submitted 0/00; Published 0/00

Rearrangement Clustering: Pitfalls, Remedies, and Applications

Sharlee Climer SHARLEE@CLIMER.US
Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130-4899, USA

Weixiong Zhang ZHANG@CSE.WUSTL.EDU
Department of Computer Science and Engineering
and Department of Genetics
Washington University in St. Louis
St. Louis, MO 63130-4899, USA

Editor:

Abstract

Given a matrix of values in which the rows correspond to objects and the columns correspond to features of the objects, rearrangement clustering is the problem of rearranging the rows of the matrix such that the sum of the similarities between adjacent rows is maximized. Referred to by various names and reinvented several times, this clustering technique has been extensively used in many fields over the last three decades. In this paper, we point out two critical pitfalls that have been previously overlooked. The first pitfall is deleterious when rearrangement clustering is applied to objects that form natural clusters. The second concerns a similarity metric that is commonly used. We present an algorithm that overcomes these pitfalls. This algorithm is based on a variation of the Traveling Salesman Problem. It offers an extra benefit as it automatically determines cluster boundaries. Using this algorithm, we *optimally* solve four benchmark problems and a 2,467-gene expression data clustering problem. As expected, our new algorithm identifies better clusters than those found by previous approaches in all five cases. Overall, our results demonstrate the benefits of rectifying the pitfalls and exemplify the usefulness of this clustering technique. Our code is available at our websites.

Keywords: Clustering, Visualization of Patterns in Data, Bond Energy Algorithm, Traveling Salesman Problem, Asymmetric Clustering

1. Introduction

> Science is organized knowledge. Wisdom is organized life.
> - *Immanuel Kant*

Clustering is aimed at discovering structures and patterns of a given data set. As a fundamental problem and technique for data analysis, clustering has become increasingly important, especially with the explosion of data on the World Wide Web and the advent of massive quantities of genomic data.

A given set of objects can be clustered in a variety of ways, depending on three criteria: the degree of granularity desired, the distance measure that is employed, and the objective that is stated as the goal for the clustering.

The degree of granularity affects clustering results. There is usually a range of values for the number of clusters k that are of interest. The desired degree of granularity is problem specific. Consider for example, clustering the population of a large geographical region. A company wishing to determine the location of a few distribution centers would desire a small k value, while a utility company may have applications requiring several thousand clusters. Once a range of values is established for k, it is frequently useful to determine clustering results for several values of k within this range and use domain knowledge to determine the best solution.

The choice of a distance measure also impacts clustering results. A distance measure is a means of quantifying the pair-wise dissimilarities between objects. Alternatively, a similarity measure is used to quantify pair-wise similarities. When objects can be accurately characterized as points residing within a metric space, the Euclidean distance is frequently employed. Distance functions are sometimes assumed to be symmetric (*i.e.*, $d(i, j) = d(j, i)$), obey the triangle inequality, and require that $d(i, i) = 0$. In this paper, we do not assume that any of these properties necessarily hold as there exist applications when effective distance measures do not obey these properties. For instance, in the realm of document clustering, the *cosine distance* is frequently employed, although this measure does not obey the triangle inequality (Steinbach et al. 2000).

Finally, the objective that is stated as the goal guides the clustering results. Clustering problems are interesting as there is no single objective that is universally applicable. Many objective functions have been proposed and used throughout the history of clustering. Some objectives optimize with respect to distances of objects to their respective cluster centers. Some base their optimizations on *diameters*, or maximum pair-wise distances of each cluster. These objectives tend to assume somewhat regular cluster configurations and can lead to undesirable results when cluster boundaries are complex as in Figure 1. Intuitive clustering using the Euclidean distance measure is shown in Figure 1(b). In this case, many objects are closer to the center of a different cluster than their own and the diameters are not minimized.

One clustering problem that has been studied extensively is the problem of identifying and displaying groups of similar objects that occur in complex data arrays (McCormick et al. 1972; Arabie and Hubert 1990; Arabie et al. 1988; Alpert 1996; Johnson et al. 2004; Torres-Velzquez and Estivill-Castro 2004). The problem can be represented as a matrix where the rows correspond to the objects to be clustered and the columns are their features. Similar objects can be identified and displayed by rearranging the rows so that the overall similarity between all adjacent objects is maximized. After rearranging the rows, the clusters are identified either manually or automatically in a second step.

This clustering problem actually consists of two objectives. The first objective is consistently used for a number of applications and requires either the maximization of the sum of similarities between adjacent rows or the minimization of the sum of distances between adjacent rows. The second objective varies in the literature, however, the general goal is to identify clusters among the rearranged objects.

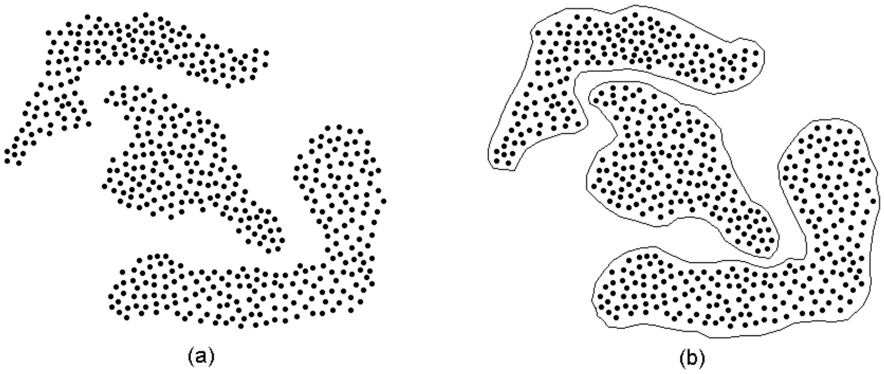

Figure 1: (a) A dataset with Euclidean distance used for the distance measure. (b) Intuitive clustering of the dataset. Many objects are closer to the center of a different cluster than their own and the diameters are not minimized.

In 1972, McCormick, Schweitzer, and White (1972) introduced the *bond energy algorithm* (BEA) which yields an approximate solution for the first objective of this clustering problem. Since that time, a "fast-growing literature" (Marcotorchino, 1987, p. 73) has appeared on this subject. This problem has been applied to a number of different applications in diverse areas, such as database design (Ozsu and Valduriez 1999), data mining (Dunham 2003), factorization of sparse matrices (Alpert 1996), matrix compression (Johnson et al. 2004), information retrieval (March 1983), manufacturing (Kusiak 1985), imaging (Kusiak 1984), marketing (Arabie et al. 1988), software engineering (Gorla and Zhang 1999), VLSI circuit design (Alpert 1996), clustering of web-users (Torres-Velzquez and Estivill-Castro 2004), shelf space allocation (Lim et al. 2004), and clustering of genes (Liu et al. 2004).

For some of these applications, it is useful to also rearrange the columns. Since the rearrangement of the columns is independent of the rearrangement of rows, the columns can be rearranged in a separate step, using the same technique that is used for the rows.

The core problem does not seem to have been given a consistent name and has been reinvented several times[1] (McCormick et al. 1972; Alpert and Kahng 1997; Johnson et al. 2004; Torres-Velzquez and Estivill-Castro 2004). It has been referred to as "structuring of matrices" (Punnen 2002), "data reorganization" (McCormick et al. 1972), "clustering of data arrays" (Lenstra 1974), "restricted partitioning" (Alpert and Kahng 1997), and "matrix reordering" (Johnson et al. 2004). Due to its nature and for the convenience of our discussion, we call this clustering problem *rearrangement clustering*.

Almost all of the existing rearrangement clustering algorithms have focused on arranging the objects to approximately maximize the overall similarity (or minimize the overall dissimilarity) between adjacent objects, while few methods have been developed to automatically identify the clusters of objects that form natural groups. An exception is the work of Alpert and Kahng (1997), in which they identified optimal partitioning for a given

1. In fact, we also reinvented it ourselves at the beginning of this research.

number of clusters k. For many rearrangement clustering algorithms, the objects are first rearranged, then a domain expert determines the cluster intervals.

Although rearrangement clustering has been extensively used for more than 30 years, there are two serious pitfalls that have been previously overlooked. The first pitfall is deleterious when the objects to be rearranged form natural clusters; which is the case for every application we have observed. The second pitfall concerns the use of the *measure of effectiveness* (ME) metric, which is employed by the bond energy algorithm.

In this paper, we first briefly summarize background material. Then we identify two pitfalls of previous approaches. In Section 4 we present techniques for rectifying these pitfalls. Section 5 describes the implementation of rearrangement clustering without the pitfalls. We summarize the results of using this implementation for four benchmark problems and a 2,467 gene expression data clustering problem in Section 6. We conclude this paper with a brief discussion. A preliminary report on this work appeared in an earlier paper (Climer and Zhang 2004).

2. Background

Given a matrix in which each row corresponds to an object and each column corresponds to a feature of the objects, rearrangement clustering is the problem of shuffling the rows around until the sum of the similarities between adjacent rows is maximized. The similarity of two objects can be measured by a similarity score defined on their features.

More formally, let P represent the set of all possible permutations of rows for a given matrix and $s(i,j)$ represent a non-negative similarity measure for objects (rows) i, j. Then an optimal permutation $p \in P$ for the given similarity measure is

$$V(P) = max\left(\sum_{i=1}^{n-1} s(i, i+1)\right) \tag{1}$$

for n objects. Conversely, given a non-negative *dissimilarity* function, $d(i,j)$, an optimal permutation $p \in P$ is

$$W(P) = min\left(\sum_{i=1}^{n-1} d(i, i+1)\right) \tag{2}$$

2.1 Bond Energy Algorithm

One of the first algorithms to tackle rearrangement clustering was the bond energy algorithm (BEA) (McCormick et al. 1972). BEA uses the measure of effectiveness (ME) in which the similarity measure for two rows, i and j, is

$$s(i,j) = \sum_{k=1}^{m} a_{ik}a_{jk} \tag{3}$$

where m is the number of features and a_{ik} is the (non-negative) kth feature of object i.[2] Hence, each element in the matrix, except those in the last row, is multiplied by the element

2. McCormick et al. used a single ME function to simultaneously quantify similarities of adjacent columns as well as adjacent rows.

directly below it, and ME is equal to the sum of these products. The intuition behind this similarity measure is that large values will be drawn to other large values, and small values to other small values, so as to increase the overall sum of the products. The term *bond energy* expresses this concept. BEA computes an approximate solution that attempts to maximize ME.

BEA has gained wide recognition and remains the algorithm of choice for a number of applications. One such use arises in manufacturing. In these applications, parts or machines with similar features are grouped into families in a process referred to as *cell formation*. Chu and Tsai (1990) compared three rearrangement algorithms for this application: rank order clustering (ROC) (King 1980), direct clustering analysis (DCA) (Chan and Milner 1982), and BEA. They ran trials for various manufacturing applications and found that BEA outperformed the other two algorithms in all of their tests.

BEA is also popular for database design. The goal here is to determine sets of attributes that are accessed by distinct sets of applications, using a process referred to as *vertical fragmentation* (Ozsu and Valduriez 1999). BEA has been promoted for this use by (Hoffer and Severance 1975) and (Navathe et al. 1984). Furthermore, BEA is included in textbooks on database design (Ozsu and Valduriez 1999) and data mining (Dunham 2003).

BEA has also been used for analyzing program structure in the field of software engineering (Gorla and Zhang 1999). The locations of all of the components, their respective calls, and the depth of nested calls all contribute to the difficulties that can be expected during the debugging and maintenance phases of a program's life. Due to the fact that these phases are generally much more expensive than the other phases, structural improvements are valuable. BEA has been used to determine the placement of components with good results (Gorla and Zhang 1999).

A recent application of BEA was the clustering of gene expression data (Liu et al. 2004). The current microarray gene expression profiling technology (Baldi and Hatfield 2002; Eisen et al. 1998) is able to examine the expressions of hundreds, thousands or even tens of thousands of genes at once. A large amount of microarray data has been collected on numerous species and organisms, ranging from microbial organisms to plants to animals. The results of a set of microarray experiments on a collection of genes under different conditions are typically arranged as a matrix of gene expression levels in real values, where the rows represent the genes to be analyzed and the columns corresponds to experimental conditions (Baldi and Hatfield 2002; Eisen et al. 1998). The objective is to identify and display clusters of genes that have similar expression patterns. BEA was shown to outperform k-means for the clustering of 44 yeast genes (Liu et al. 2004).

2.2 Traveling Salesman Problem

It has been pointed out that rearrangement clustering is equivalent to the Traveling Salesman Problem (TSP) and can be solved to *optimality* by solving the TSP (Lenstra 1974; Lenstra and Kan 1975). The TSP for n cities is the problem of finding a tour visiting all the cities and returning to the starting city such that the sum of the distances between consecutive cities is minimized. In other words, the TSP is to find a cyclic permutation of the cities so that the total distance of adjacent cities under the permutation is minimized. It is well known that TSP is NP-hard (Karp 1972).

The mapping from a rearrangement clustering problem instance to a TSP instance is straightforward (Lenstra 1974; Lenstra and Kan 1975). We first view each object as a city and transform the dissimilarity between two objects to the distance between the corresponding cities. The TSP tour, which must have the minimum distance among all complete tours, is an optimal rearrangement of the objects with the minimum dissimilarity. (We use the words *distance* and *dissimilarity* synonymously in this paper.) Thus, the TSP is the same problem as finding an optimal permutation p, except that the TSP finds a cycle through the cities and rearrangement clustering finds a path.

This discrepancy can easily be rectified by adding a *dummy city.* A dummy city is an added city whose distance to each of the other cities is equal to a constant C. The location of the dummy city is the optimal point for breaking the TSP cycle into a path (Lenstra and Kan 1975). The TSP path is defined as the TSP tour with the dummy city and its two incident edges excluded. The length of this path is equal to the length of the tour minus $2C$. Following are two critical observations on the above extended TSP.

Lemma 1 *The direct distance between the two cities that are separated by the dummy city is greater than or equal to any of the distances between adjacent pairs of cities on the TSP tour, and the total distance of the TSP path is the smallest possible.*

Proof: We prove the first part of the Lemma by contradiction. Assume that the distance $d(x, y)$ between an adjacent pair of cities, x and y, on the TSP tour T is greater than the direct distance $d(u, v)$ of the two cities, u and v, which are spanned by the dummy city. That is, $d(u, v) < d(x, y)$. Then we can directly connect cities u and v, and insert the dummy city between cities x and y, with a net difference of $d(u, v) - d(x, y) < 0$ to the final tour length. This contradicts the fact that T is a minimum-distance complete tour.

We prove the second part of the Lemma by contradiction also. Assume the length of the TSP path is D and that there exists a path with a length D' where $D' < D$. A cycle that includes the dummy city can be constructed using the new path and its length is $D' + 2C$. This cycle is a feasible solution to the original TSP, but has a length that is shorter than the original TSP solution of $D + 2C$. This contradicts the fact that the original cycle has the minimum possible length. $\square$

2.3 Restricted Partitioning

A well-known approximation algorithm for solving the Euclidean TSP was introduced by Karp (1977) and uses the rule of thumb that every city within the current cluster is visited before moving out of the cluster. This work was cited two decades later and it was proposed that the "inverse" of Karp's algorithm be used to determine clusters *i.e.*, solve the TSP to find the clusters (Alpert and Kahng 1997). In other words, Alpert and Kahng reinvented rearrangement clustering and referred to it as *restricted partitioning* (RP). They took rearrangement clustering a step further, however, as they introduced an algorithm for automatically determining the locations of cluster boundaries for a given TSP solution and a given number of clusters k. This algorithm computes the boundaries that will yield a set of clusters in which the largest diameter is as small as possible. This partitioning algorithm is based on dynamic programming and runs in $O(kn^3)$ time when applied after solving a TSP tour and $O(kn^2)$ time when applied after solving a TSP path.

Alpert and Kahng applied rearrangement clustering to various problems, including clustering of flower types and clustering cities according to their average temperatures throughout the year (Alpert and Kahng 1997). However, the main focus of their work was on partitioning circuits for use in the computer-aided design of VLSI circuits (Alpert 1996).

2.4 Matrix Reordering

Rearrangement clustering was recently reinvented by David Johnson et al. and referred to as *matrix reordering* (Johnson et al. 2004). This work presents a lossless compression strategy for effective storage and access of large, but sparse, boolean matrices on disk. The columns of these matrices are rearranged so as to bring together the one's in the rows. In their paper, the problem was identified as a TSP. This work was demonstrated by compressing matrices within the domains of interactive visualization and telephone call data.

As with Alpert and Kahng's work, rearrangement clustering was taken a step further in Johnson et al.'s paper. For the problems they addressed, finding even an approximate solution for the TSP was obstructed as many of these problems were too large to fit into main memory. To address this obstacle, they devised a multi-faceted approach that blends classical TSP heuristics with instance-partitioning and sampling. Their approach resulted in significant improvements in access time as well as compression.

2.5 Rearrangement Clustering

Just prior to the work done by Johnson et al., we reinvented rearrangement clustering ourselves (Climer and Zhang 2004). We were motivated by the need to cluster gene expression data and inspired by the work of Eisen et al. (1998). In their work, Eisen et al. clustered a 2,467 yeast gene set using hierarchical clustering, then arranged the results in a linear order, creating a matrix in which each row corresponded to a gene and each column to an experimental condition. After creating this matrix, they examined it and manually determined cluster boundaries.

Given that the objective was to derive a matrix from which clusters could be identified, we set out to optimize such a matrix. That is when we reinvented rearrangement clustering and identified it as the TSP. However, we soon realized there was a flaw in our approach. This pitfall is described in the next section.

3. Pitfalls

Although rearrangement clustering has been extensively studied and used over the last three decades, there is a serious flaw in previous approaches when applied to data that falls into natural clusters. Consider the example illustrated in Figure 2, where objects have only two features (their horizontal and vertical coordinates) and the dissimilarity between objects is the Euclidean distance. When these objects are rearranged according to the objective in (2), the large cluster on the bottom is broken in half and placed at each end of the ordering. Although objects x and y are very similar, they are separated by 16 objects in two different clusters. We use this simple example as the optimal solution is obvious. However, it is clear that in general, clusters may be broken in pieces in order to minimize the "jumps" to adjacent clusters.

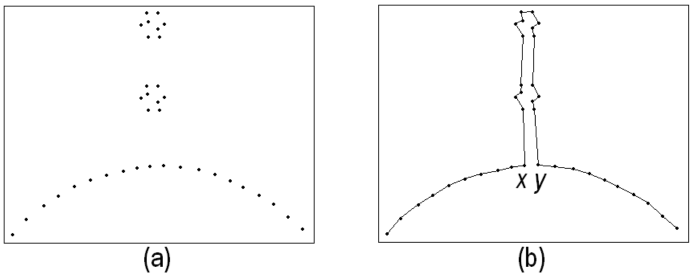

Figure 2: (a) Three clusters. (b) The TSP path specifying the optimal rearrangement. Although x and y are very close, their placement in the rearrangement is far apart.

0	0	0		0	0	0		0	1	0
1	0	1		0	1	0		0	0	0
0	1	0		1	0	1		1	0	1
	(a)				(b)				(c)	

Table 1: Three rearrangements of a matrix with an optimal ME. (a) One pair, (b) two pairs, (c) three pairs of zeros are aligned.

When natural clusters occur, the inter-cluster distances are much greater than the intra-cluster distances. Therefore, the sum of distances between adjacent objects in objective (2) is dominated by the inter-cluster distances. The rearrangement may skew itself in order to minimize these large distances. In the next section, we propose an alternative objective function that addresses this defect and present a technique for resolving this new objective.

The second pitfall applies to the measure of effectiveness (ME) that is used by BEA. ME uses the similarity measure that is defined in equation (3). Two problems associated with ME are that it can fail to ascertain the quality of clustering of non-maximal values and it tends to push small values to the top and bottom of the rearranged matrix. Consider the following examples. Table 1 shows three arrangements of a binary matrix that have the same ME = 0, which is the highest value possible. ME fails to distinguish between the levels of clustering of the pairs of zeros. This behavior is not limited to zeros. Table 2 shows three arrangements of a ternary matrix. The first two have ME values of 16, which is optimal. However, the first fails to bring together any of the three identical rows, each containing all ones. Moreover, note how the small values are pushed to the top and bottom of the array. The third arrangement is more likely to be of use for most applications, but it has a sub-optimal ME value of 15.

0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0
1	2	1	1	2	1	1	1	1
1	1	1	2	1	1	1	1	1
2	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	2	1
0	0	0	0	0	0	2	1	1
	(a)			(b)			(c)	

Table 2: Three rearrangements of a matrix. (a) ME = 16, which is optimal. The three identical rows of ones are all separated. (b) ME = 16. Two of the rows of ones are adjacent, but the third is separated by two intervening rows. (c) A clustering that would probably be preferred, but ME = 15.

4. Remedies

The second pitfall can be easily rectified by using a different similarity measure. There are a number of similarity measures that are available, including simple Euclidean distances and somewhat more complicated correlation coefficients. In general, the measure that is used can have a profound effect on clustering results and should be selected to suit the problem that is addressed.

We now turn our attention to the first pitfall. A remedy to this pitfall is to omit the inter-cluster distances from the sum in objective (2). We redefine our objective as follows:

$$W(p,k) = min\left(\sum_{i=1}^{k}\sum_{j=u_i}^{v_i-1} d(j,j+1)\right) \tag{4}$$

where u_i is the first item and v_i is the last item of cluster i, and k is the number of clusters. The inner summation of objective (4) is the sum of distances between adjacent rows within a cluster and the outer summation is over all the clusters. In this way, we minimize the intra-cluster distances while disregarding the inter-cluster distances. The inter-cluster distances will assume whatever values best suit the minimization of intra-cluster distances.

This revised problem can be solved using the TSP with a twist. The key to solving this problem lies in Lemma 1. What if we introduce k dummy cities to the TSP representation of the clustering problem? Just as one dummy node cuts the TSP cycle into a path, these dummy cities virtually cut the tour into k segments and form the cluster borders. The distances between pairs of dummy cities are set to infinity, to ensure that no two dummy cities are adjacent on the tour. After this "TSP+k" problem is solved, the dummy cities and their incident edges are removed and replaced by cluster borders. The lengths of the edges that span the borders, or *borderline* edges, are not of any consequence in the solution of TSP+k. In this way, the TSP+k solution optimizes the intra-cluster distances, while disregarding inter-cluster distances. As a bonus, the cluster boundaries are automatically identified.

Theorem 2 *When there exist k dummy cities, the sum of the lengths of the k paths that are defined by the TSP+k tour is minimized, and every edge in these paths has a distance that is no longer than any of the resulting k borderline edge lengths.*

Proof: No two dummy cities are adjacent on the TSP+k tour, as the distance between them is infinity. Therefore, every TSP+k tour has $2k$ edges of cost C that are adjacent to the dummy cities. The rest of the proof is similar to the proof of Lemma 1. □

In Figure 3, an example of the use of this new objective function is shown. A set of color samples are rearranged, using the intensities of their red, green, and blue components as their features. BEA finds a suboptimal solution as shown in the figure. Solving the TSP with objective (2) leads to splitting the large color cluster in half and inserting the gray color cluster in order to reduce the inter-cluster distance. Note that the color immediately above the gray cluster is very similar to the color immediately below the gray cluster, yet they are far apart in the rearrangement. Moreover, none of the gray colors separating them are nearly as similar to either of them as the two are to each other. This solution is optimal for objective (2). Restricted partitioning (RP) automatically identifies the cluster boundaries as shown in Figure 3(d). RP yields the same linear ordering as TSP+k with $k = 1$. The partitioning minimizes the maximum diameter of the clusters. Notice that this goal splits the gray colors between the two clusters. Finally, by using the new objective in (4) and adding a second dummy city, the inter-cluster distance is ignored and the two clusters are correctly formed as shown in Figure 3(e).

Theorem 2 guarantees the optimality of identifying k clusters for a given k, based on the objective function (4). Assuming that a range of k values is specified, determining the best value for k within this range is the next consideration.

Theorem 3 *Let*

$$d_{mean} = \frac{\sum_{i=1}^{k} \sum_{j=u_i}^{v_i-1} d(j, j+1)}{(n-k)} \tag{5}$$

As k increases, d_{mean} is non-increasing.

Proof: d_{mean} is the average intra-cluster distance. In general, as k increases, the membership of clusters may be rearranged to provide the current optimal solution. Let us consider the special case in which the number of clusters is increased from k to $k+1$ and the only change in the TSP tour is that a single edge is replaced by two edges with costs C and the new dummy node. From Theorem 2, we know that the deleted edge must have the maximum distance. Thus, the average distance of the edges cannot increase with this change. Since the TSP finds the minimum tour distance, this property holds when the tour undergoes more than just this one minor change. Therefore, the average distance within clusters is non-increasing. □

The TSP+k algorithm guarantees an optimal rearrangement clustering for a given k. However, as shown by Theorem 3, we must consider desirable qualities other than average intra-cluster distance when determining the best value for k. One approach to handling this problem is to run the algorithm for each value of k in the desired range and use problem-specific information to determine the best clustering result. Another approach is based on the observation that a clustering in which the clusters are well-defined will tend to have large distances between clusters. Therefore, an analysis of the changes in inter-cluster distances may be useful in determining the best k.

When using TSP+k, the resulting clusters are randomly ordered. For some of our experiments, we applied TSP to the border cities to determine an ordering of the clusters. The resulting ordering minimizes the distances between clusters. While this is not necessary for identifying the clusters, it yields useful information about the average distance between clusters, may aid the evaluation of various k values, and may be advantageous for displaying the clustering result. It is also useful if it is desirable to merge small clusters in a post-processing step.

5. Implementation

Our code is composed of two programs and is available at *http://www.climer.us* or *http://www.cse.wustl.edu/~zhang/projects/software.html*. The first program converts a data matrix into a TSP problem, and the second rearranges the rows of the data according to the TSP solution. Any TSP solver can be used. Our method is usable for large clustering problems thanks to recent advances in TSP research.

The TSP has been extensively studied for many decades. A plethora of papers have been written, books have been published (Gutin and Punnen 2002; Lawler et al. 1985), and websites have been devoted to this problem (Cook web; Moscato web; Johnson web).

There has been a vast amount of research devoted to solving TSPs to guaranteed optimality. For all of our experiments, we used Concorde (Applegate et al. 2001), an award winning TSP solver that has successfully solved a record 24,978-city TSP instance to optimality. The Concorde code is publicly available at *http://www.tsp.gatech.edu//concorde.html*.

There are many applications in which computation time is critical. Fortunately, a great deal of research has been devoted to quickly finding high-quality approximate solutions for the TSP, yielding a wealth of available code (Lodi and Punnen 2002). These implementations vary drastically in running time and quality of solutions. The fastest compute solutions almost as quickly as the input can be read. Others run more slowly but yield more accurate solutions. For instance, Helsgaun's method (2000), which is based on the Lin-Kernighan heuristic (Lin and Kernighan 1973), has produced the optimal solution for every optimally solved problem Helsgaun has obtained, including a 15,112-city TSP instance. It has also improved upon the best known solutions for a number of large-scale instances, including a 1,904,711-city problem. Useful comparisons of time versus quality for a number of approximate algorithms are available (Johnson and McGeoch 2002; Arora 2002).

In some cases there are an extremely large number of objects that need to be clustered. Propitiously, TSP research has explored the problem of solving very large instances. For example, Johnson et al. presented an algorithm for solving TSP instances that are too large to fit into main memory (Johnson et al. 2004).

In some applications, there may exist a distance function that is not strictly symmetric. That is, $d(i, j)$ may not necessarily be equal to $d(j, i)$. For example, the affinities between amino acid sequences are frequently asymmetric due to different lengths of the sequences and/or asymmetries in the amino acid substitution matrix. In these cases, TSP+k is still viable. Instead of solving a symmetric TSP, an asymmetric TSP (ATSP) would be computed. There are a number of approximation algorithms for the ATSP and comparisons of these algorithms are available (Johnson et al. 2002). Optimal solutions can be found using branch-and-bound (Carpaneto et al. 1995), branch-and-cut (Fischetti et al. 2002), or cut-and-solve (Climer and Zhang 2006). Furthermore, symmetric TSP (STSP) codes such as Concorde could be used by converting the ATSP instance into an STSP instance. One way to make this conversion is the *2-node* transformation (Jonker and Volgenant 1983), in which the number of cities is doubled.

6. Experimental Comparisons and Applications

In this section, we describe four benchmark problems as well as a 2,467-gene expression data clustering problem. The benchmark problems were previously solved using BEA with the ME metric. The clusters were manually identified by domain experts. We present comparisons with TSP+k using the ME metric. We did not compare these results with restricted partitioning (RP). Such a comparison would be misleading as RP minimizes with respect to cluster diameters. In the final part of this section, we present the results of clustering a yeast gene data set. Yeast genes have been extensively studied and functionally related groups have been identified. This research allows objective evaluation of cluster quality. We used these evaluations to compare results from hierarchical clustering, RP, and TSP+k.

6.1 Testbed

Four examples from diverse application domains have been previously presented in the literature. The first three were compared in (McCormick et al. 1972; Lenstra and Kan 1975). They include an airport design example, an aircraft types and functions example, and a marketing applications and techniques example. The fourth example was used in (March 1983) for clustering personnel database records.

We also tackled rearrangement clustering of a large set of gene expression data. The dataset consists of 2,467 genes in the budding yeast *Saccharomyces cerevisiae* that were studied during the diauxic shift (DeRisi et al. 1997), mitotic cell division cycle (Spellman et al. 1998), sporulation (Chu et al. 1998), and temperature and reducing shocks (P.T. Spellman, P.O. Brown, and D. Botstein, unpublished results), yielding 79 measurements that are used as the features for the genes. These genes were previously clustered (Eisen et al. 1998) and the data is available at the PNAS website (*http://www.pnas.org*).

6.2 Results for Benchmark Problems

In this section, we first compare the old and new objective functions for four problems that have been presented in (McCormick et al. 1972; Lenstra 1974; March 1983). We based our comparisons on the quality of the individual clusters that are identified. Since McCormick et al. and March manually determined clusters based on the ME metric, we also used ME and compared cluster quality using the ME metric for these four problems.

The clusters identified by McCormick et al. (1972) do not strictly partition the objects. There is some overlapping and some objects are left unclustered. Overlapping of clusters is not allowed in most applications and is not addressed by our new objective. For these reasons, our comparisons are based on non-overlapping results.

The marketing example is shown in Figure 4. The rows represent applications and the columns represent various techniques. This is a binary matrix, where a one indicates that a technique has been shown to be useful for an application and a zero indicates that it has not been useful. This is the only example we present in which it is desirable to find clusters for both the columns and the rows. The ME for the entire matrix is equal whether approximately solved by BEA or optimally solved as a TSP with $k = 1$. When clustering was performed on the techniques (columns), TSP+k with $k = 17$ identified the same three clusters that were identified by McCormick *et al.* (1972). When TSP+k was used to cluster the applications (rows), it identified clusters with two, three, and four elements, respectively. Notice that McCormick *et al.* identified three clusters, which overlapped for the applications (rows) clustering. We used the four- and three-element clusters that were not overlapping for comparisons. The four-element cluster was the same for both algorithms. However, the three-element clusters differed by one application, *i.e.* BEA grouped together 'sales forecasting', 'brand strategy', and 'advertising research' while TSP+k substituted 'pricing strategy' for 'sales forecasting'. Computing the ME for the three applications yielded a value of 8 for BEA and 10 for TSP+k, revealing that, based on ME, the cluster identified by TSP+k is of higher quality.

The airport design example was presented in (McCormick et al. 1972) to demonstrate how BEA can be used for problem decomposition, reducing a large project into a set of small projects with minimal interdependency. The values in the matrix were set to 0, 1, 2, or 3 to indicate no, weak, moderate, or strong dependencies respectively. Figure 5 shows the results for this data. The ME for the entire matrix is 577 for BEA and improved to 580 for TSP with $k = 1$. McCormick *et al.* (1972) identified eight clusters, with three pairs of clusters overlapping by one object. To make comparisons, we eliminated these overlaps by including the overlapped object in only one cluster, the one that increased the ME value the most. In order to compare the quality of the clusters, we only considered the intra-cluster similarities and ignored the similarities between adjacent clusters. The ME for each cluster was computed and the sum of ME values for the eight clusters is 464 for BEA and 503 for TSP+k with $k = 8$, yielding an improvement in the quality of the clusters.

Figure 6 shows the results for rearrangement clustering of aircraft types based on their functions. Values in the matrix were set from zero to two reflecting the extent that the aircraft can perform the function. The ME for the entire matrix is 1930 for BEA and 1961 for TSP with $k = 1$. The clusters identified by McCormick *et al.* (1972) had substantial overlapping. We compared the two largest clusters, containing 24 and 14 aircraft, respectively. The two largest clusters for TSP+k with $k = 17$ also contained 24 and 14 aircraft. The sum of the ME values for the two clusters is 1545 for BEA and is 1616 for TSP+k.

Figure 7 shows the results for the personnel database records example from (March 1983). The values in this matrix range from one to one hundred. The ME for the entire matrix is 1,791,870 for BEA and 1,836,260 for TSP with $k = 1$. March identified six clusters with no overlapping. The sum of the ME values for these clusters using BEA is 1,533,034 and for TSP+k with $k = 6$ is 1,645,207.

6.3 Gene Expression Data

In this section we compare rearrangement clustering methods for yeast gene expression data. Yeast genes have been extensively researched and annotated, allowing objective evaluation of the quality of clusters found by each method.

6.3.1 Metrics

In our gene clustering tests, we used the Pearson correlation coefficient (PCC) for the similarity measure. The PCC is defined as follows:

$$s(x,y) = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{\left(\sum X^2 - \frac{(\sum X)^2}{N}\right)\left(\sum Y^2 - \frac{(\sum Y)^2}{N}\right)}} \tag{6}$$

where X and Y are the feature vectors for genes x and y, respectively, and N is the number of features for which both x and y have data tabulated. PCC has been extensively used for gene expression data clustering and was used for comparisons of gene clustering algorithms in (Shamir and Sharan 2002). After finding the similarities, we scaled and applied an additive inverse to translate the similarities to nonnegative integral distances.

To objectively evaluate the performance of the various algorithms on gene expression data, we used Gene Ontology (GO) Term Finder (*http://www.yeastgenome.org/*), a tool for finding functionally related groups of yeast genes in a given cluster. This tool calculates a p-value that indicates the likelihood of observing a group of u genes with a particular functional annotation in a cluster containing v genes, given that M genes have this annotation in the total population of N genes. More specifically, the p-value is equal to

$$1 - \sum_{j=0}^{u-1} \frac{\binom{M}{j}\binom{N-M}{v-j}}{\binom{N}{v}} \tag{7}$$

Notice that the size of the cluster is reflected in calculating the p-value. For instance, if a small and a large cluster both contain a group of u genes with a particular functional annotation, the p-value will be greater for the group in the large cluster as the probability of finding u genes with the given functional annotation is greater in a larger cluster.

6.3.2 Results for Gene Expression Data

In this section, the results of using three different algorithms for clustering the 2,467 yeast gene dataset are presented. The first algorithm uses a hierarchical technique and was presented by Eisen et al. (1998). After applying hierarchical clustering, the results were illustrated in a linear fashion and ten clusters were identified by a domain expert. The identification of clusters was the same as is commonly used in rearrangement clustering. However, the rearrangement of the rows was not based on finding maximum similarity between adjacent rows. Out of the 2,467 genes, 263 were selected for the ten clusters that were identified. It was observed by Eisen et al. that each cluster contained genes that are functionally related.

	(a)	(b)	(c)	(d)	(e)	(f)
TSP+k	44	56	100	13	129.6	40
RP	44	0	44	13	101.4	28
TSP+k	77	123	200	13	81.8	44
RP	77	0	77	16	42.2	34
TSP+k	109	191	300	16	63.8	41
RP	109	1	110	18	33.1	39
Eisen et al.	10	-	-	9	26.3	49

Table 3: Results for 2,467 yeast gene clustering where "good" functional groups are defined as those with p-values with orders of magnitude $\leq 10^{-7}$. (a) Number of non-singleton clusters. (b) Number of singleton clusters. (c) Value of k used. (d) Number of clusters found containing "good" functional groups. (e) Average size of these "good" clusters. (f) Number of "good" functional groups.

We ran TSP+k with $k = 100$, $k = 200$, and $k = 300$ on the 2,467-gene data set. Our results are compared with restricted partitioning (RP) (Alpert and Kahng 1997) and the results from Eisen et al. (1998). We adjusted the k value for RP so as to yield the same number of non-singleton clusters for comparisons. GO Term Finder (*http://www.yeastgenome.org/*) was run for each cluster found in each trial and on the ten clusters identified by Eisen et al. Functional groups found with p-values having orders of magnitude less than or equal to 10^{-7} were designated as "good" functional groups. Tables 3 and 4 contain the results of these trials. Figure 8 displays the reordered matrices for TSP+k.

An interesting result of the TSP+k tests was the large number of singletons, as listed in Table 3. In all cases, more than half of the clusters contained singletons. Yet there was not a dominance of clusters containing only two or three genes. For instance, there were only six clusters containing two or three genes when $k = 100$. However, that trial had 56 singletons. Gene expression data is notoriously noisy, so many of the singletons that were found may correspond to outliers in the data. This result suggests that TSP+k may be useful for identifying outliers.

For all the values of k that we tested, the rearrangement clustering algorithms found more "good" clusters than the nine found by Eisen et al. RP found more "good" clusters than TSP+k for the two larger trials. However, the TSP+k clusters were larger in all three trials and a greater number of the 2,467 genes were placed into meaningful clusters. Note that the p-value essentially reflects the *concentration* of related genes within a cluster. Consequently, for a fixed p-value and a particular functional relationship, a larger cluster contains more of these related genes than a smaller cluster.

Table 3 lists the number of "good" functional groups found. TSP+k found more of these groups than RP for each run. Eisen *et al.* found more groups than any of the rearrangement clustering trials. However, the rearrangement clustering algorithms found more distinct functional groups than Eisen et al. when the results from the three trials are combined. Table 4 lists the combined results. Eisen et al. found 49 "good" functional groups. 25 of these groups were missed by RP and 18 were missed by TSP+k. RP found a total of 48 distinct functional groups. 24 of these were missed by Eisen et al. and 11

	Total number of distinct groups	Number missed by Eisen et al.	Number missed by RP	Number missed by TSP+k
Eisen et al.	49	-	25	18
RP	48	24	-	11
TSP+k	61	37	33	-

Table 4: Total number of distinct "good" functional groups found by each algorithm. For each algorithm, the number of groups missed by the other algorithms are shown.

were missed by TSP+k. Finally, TSP+k found 61 distinct functional groups. Eisen et al. missed 37 of these and RP missed 33. Some of these functional groups were related and appeared in the same cluster. For example, in all but one trial, TSP+k identified a "good" cluster containing functionally related groups of genes involved in carbohydrate transporter activity and six related functions. All seven of these functional groups were overlooked by both RP and Eisen et al.

Tables listing the functional groups for each trial can be found on the web at *http://www.climer.us/cluster/TSPX.htm* and *http://www.climer.us/cluster/RPX.htm*, where X is replaced by the value of k. The results for (Eisen et al. 1998) can be found at *http://www.climer.us/cluster/eisen.htm*.

6.3.3 Changes in Domain Knowledge

About a year ago, we ran GO Term Finder on the clusters found by Eisen et al. and those found using TSP+k. The results are listed in Table 5. It can be expected that a number of additional genes have been annotated during the year, resulting in changes in p-values. In other words, the clusters themselves have not changed during the year, but some of the p-values have, due to additional information found by other means. For Eisen et al., the number of "good" functional groups increased by one. For the three TSP+k trials, the number of "good" functional groups increased by one, two, and three groups respectively. For TSP+k, the number of "good" clusters remained the same for $k = 100$ and $k = 300$ and increased from 12 to 13 for $k = 200$. However, for Eisen et al., the number of "good" clusters decreased from 10 to 9.

Eisen's group published their work in 1998. If they were to redetermine the clusters today, they could use the additional information that has been found experimentally since that time to improve their results. RP and TSP+k do not rely on prior knowledge of functionally related groups to determine the clusters. If they were run in 1998, they would have yielded the same clusters as they do today. Indeed, if they were run before any knowledge of yeast functions was realized, they still would have produced these same clusters.

When using domain experts to determine cluster boundaries, the quality of the results is dependent on the current knowledge of the experts. As more knowledge is acquired in a domain, the clustering results found previously may become obsolete. Automated methods do not rely on current domain knowledge and consequently do not suffer from this antiquation. Moreover, automated methods can be used when there is little or no *a priori*

	k	old number of clusters	new number of clusters	old number of groups	new number of groups
TSP+k	100	13	13	39	40
TSP+k	200	12	13	42	44
TSP+k	300	16	16	38	41
Eisen et al.	-	10	9	48	49

Table 5: Comparisons of current GO Term Finder results with those found a year ago. This table lists the number of clusters containing "good" functional groups and the total number of "good" functional groups.

knowledge or when the use of domain experts is impractical. The latter case can occur when the cost of a domain expert is too high or the number of objects is too large.

6.4 Computation Time

The time required to run either TSP+k or RP depends on the algorithm used to solve the TSP. Fast approximate TSP solvers can be used when computation time needs to be minimized.

In the experiments presented in this paper, Concorde was used to solve each instance optimally. (During these tests Concorde aborted early several times and required restarting.) For the 2,467-gene problem, the computation time ranged from 3 to 22 minutes on an Athlon 1.9 MHz processor with two gigabytes memory. An advantage of RP over TSP+k is that a single TSP solution can be used for various values of k. Each partitioning of the TSP path runs in $O(kn^2)$ time.

BEA and hierarchical clustering arrange objects quickly, but both techniques require a domain expert to identify the cluster boundaries. CPU time has become surprisingly inexpensive and a very large number of CPU hours would be equivalent in value to a single hour of a domain expert's time. Moreover, identifying clusters manually requires a fair amount of time. We can be certain that Eisen's group spent substantially more time identifying cluster boundaries than our computer spent solving TSPs. On the other hand, a domain expert simultaneously determines the number of clusters k while identifying cluster boundaries. For the 2,467-gene data set, we arbitrarily set k equal to 100, 200, and 300. Multiple solutions can be advantageous when attempting to maximize the number of functionally related groups as in this example. However, a single solution is frequently desired in many domains. Future work to automatically determine the "best" clustering for a set of k values would maximize the efficiency of rearrangement clustering for these cases. This determination could be based on inter-cluster distances (as discussed in Section 4) and/or other qualities of the clustering results.

7. Discussion

In this section, we examine a couple of considerations that may arise when using rearrangement clustering.

7.1 Number of Features

An interesting property of TSP+k is that the number of features has little impact on the computation time. More features may increase the time required to compute the distances between cities. However, the time required to actually solve the TSP is not directly dependent on the number of features.

While the number of features has little effect on the computation time, it may have bearing on the quality of the results. When the number of features is much greater than the number of objects, *transitivity* of the similarity measure might not be upheld. The transitive property requires that if object x is similar to object y, and y is similar to object z, then x and z are similar. In the previous work we have examined, transitivity is apparently assumed, though it is not explicitly stated. Given an appropriate similarity measure, transitivity might be expected when the number of objects is large in comparison to the number of features. However, care should be used in applying rearrangement clustering when the converse is the case.

7.2 Linearity Requirement

Rearrangement clustering requires a linear ordering of objects. Visualization of complex data is enhanced by arranging objects in this manner (Eisen et al. 1998; McCormick et al. 1972). For some applications, the objects are actually placed in a linear manner, such as shelf space allocation (Lim et al. 2004). However, for many of the problems that have been previously solved using rearrangement clustering, linearity is not inherently necessary for identifying clusters.

When using rearrangement clustering, there is no quality assurance requiring that the diameters of clusters are less than a given value. This may be a concern for large clusters, in which the first and last objects may be quite dissimilar. On the other hand, this property may be advantageous when it is useful to identify elongated, but contiguous, clusters or irregularly shaped clusters as in Figure 1. Although rearrangement clustering requires the objects be linearly ordered, it doesn't suffer from the drawbacks that can arise when the objective is based on minimizing diameters or minimizing distances of objects from the centers of their respective clusters. In essence, rearrangement clustering yields solutions that tend to be contiguous as each object is a relatively short distance from at least one other object in the same cluster.

Furthermore, the addition of k dummy cities appears to increase the viability of the use of rearrangement clustering for general clustering problems. To gain some intuition about this, consider a traveling salesman who is given k free "jumps" and is required to visit n cities that fall into k distinct clusters. It is reasonable to expect that he will frequently find it most economical to use the free jumps for the long distances between clusters as opposed to using them for intra-cluster hops. When this is the case, TSP+k will correctly identify the k clusters. For example, when TSP+k with $k = 3$ is applied to the example in Figure 2, the three clusters are correctly identified.

As a final note, previous experiments have shown that rearrangement clustering, despite its pitfalls and linearity requirement, has outperformed non-linear-ordering clustering algorithms for applications that do not require linear ordering (Alpert 1996; Alpert and Kahng 1997; Liu et al. 2004).

8. Conclusion

Rearrangement clustering has been extensively used in a variety of domains over the last three decades. Yet, the previous approaches have overlooked two serious pitfalls: the summation in the objective function is dominated by inter-cluster distances and the ME metric can fail to appropriately quantify the quality of clustering. These pitfalls can be remedied by using the TSP+k algorithm and an alternate metric. As a bonus, TSP+k provides automatic identification of cluster boundaries.

By translating rearrangement clustering into the TSP, it is possible to take full advantage of the wealth of research that has been invested in optimally and approximately solving TSPs. Generally speaking, BEA is a relatively simple greedy approximation when compared to highly-refined TSP solvers. Moreover, rearrangement clustering can be solved *optimally* for many problems using TSP solvers such as Concorde (Applegate et al. 2001), as illustrated by arranging 2,467 genes in this paper.

Rearrangement clustering has been embraced in many diverse applications. Our new ability to overcome previous pitfalls should result in an even greater usefulness of this popular clustering technique. This usefulness is further enhanced by the fact that TSP+k does not require a domain expert to identify cluster boundaries, thus enabling its use in domains that are not well understood or when experts are unavailable.

Acknowledgments

This research was supported in part by NDSEG and Olin Fellowships and by NSF grants IIS-0196057, ITR/EIA-0113618, and IIS-0535257. We extend thanks to anonymous reviewers of the current paper as well as an earlier version (Climer and Zhang 2004) who provided valuable comments and insights. We also thank David Applegate, Robert Bixby, Vašek Chvátal, and William Cook for the use of Concorde and Charles Alpert and Andrew Kahng for use of RP. Finally, we thank Michael Eisen for a useful discussion.

References

C. J. Alpert. *Multi-way Graph and Hypergraph Partitioning.* PhD thesis, UCLA, Los Angeles, CA, 1996.

C. J. Alpert and A. B. Kahng. Splitting an ordering into a partition to minimize diameter. *Journal of Classification*, 14:51–74, 1997.

D. Applegate, R. Bixby, V. Chvátal, and W. Cook. TSP cuts which do not conform to the template paradigm. In M. Junger and D. Naddef, editors, *Computational Combinatorial Optimization*, pages 261–304. Springer, 2001.

P. Arabie and L.J. Hubert. The bond energy algorithm revisited. *IEEE Trans. Systems, Man, and Cybernetics*, 20(1):268–74, 1990.

P. Arabie, S. Schleutermann, J. Daws, and L. Hubert. Marketing applications of sequencing and partitioning of nonsymmetric and/or two-mode matrices. In W. Gaul and M. Schader, editors, *Data Analysis, Decision Support, and Expert Knowledge Representation in Marketing*, pages 215–24. Springer Verlag, 1988.

S. Arora. Approximation algorithms for geometric TSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*. Kluwer Academic, Norwell, MA, 2002.

P. Baldi and G.W. Hatfield. *DNA Microarrays and Gene Expression*. Cambridge University Press, 2002.

G. Carpaneto, M. Dell'Amico, and P. Toth. Exact solution of large-scale, asymmetric Traveling Salesman Problems. *ACM Trans. on Mathematical Software*, 21:394–409, 1995.

H. M. Chan and D. A. Milner. Direct clustering algorithm for group formation in cellular manufacturing. *Journal of Manufacturing Systems*, 1:65–74, 1982.

S. Chu, J. L. DeRisi, M. B. Eisen, J. Mulholland, D. Botstein, P. O. Brown, and I. Herskowitz. The transcriptional program of sporulation in budding yeast. *Science*, 282: 699–705, 1998.

S. Climer and W. Zhang. Take a walk and cluster genes: A TSP-based approach to optimal rearrangement clustering. In *21st International Conference on Machine Learning (ICML'04)*, pages 169–176, Banff, Canada, July 2004.

S. Climer and W. Zhang. Cut-and-solve: An iterative search strategy for combinatorial optimization problems. *Artificial Intelligence*, 170:714–738, June 2006.

W. Cook. Traveling Salesman Problem. *http://www.tsp.gatech.edu*, web.

J. L. DeRisi, V. R. Iyer, and P. O. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278:680–686, 1997.

M. H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice-Hall, Upper Saddle River, NJ, 2003.

M.B. Eisen, P.T. Spellman, P.O Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. of the Natl. Acad. of Sciences*, 95(25):14863–8, 1998.

M. Fischetti, A. Lodi, and P. Toth. Exact methods for the Asymmetric Traveling Salesman Problem. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*. Kluwer Academic, Norwell, MA, 2002.

N. Gorla and K. Zhang. Deriving program physical structures using bond energy algorithm. In *Proc. 6th Asia Pacific Software Engineering Conference*, 1999.

G. Gutin and A. P. Punnen. *The Traveling Salesman Problem and its variations*. Kluwer Academic Publishers, Norwell, MA, 2002.

H. A. Hoffer and D. G. Severance. The use of cluster analysis in physical data base design. In *Proceedings of the 1st International Conference on Very Large Data Bases*, pages 69–86, September 1975.

D. S. Johnson. 8th DIMACS implementation challenge. *http://www.research.att.com/~dsj/chtsp/*, web.

D. S. Johnson, G. Gutin, L. A. McGeoch, A. Yeo, W. Zhang, and A. Zverovich. Experimental analysis of heuristics for the ATSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*. Kluwer Academic, Norwell, MA, 2002.

D. S. Johnson, S. Krishnan, J. Chhugani, S. Kumar, and S. Venkatasubramanian. Compressing large boolean matrices using reordering techniques. In *30th Int. Conf. on Very Large Databases (VLDB)*, pages 13–23, 2004.

D. S. Johnson and L. A. McGeoch. Experimental analysis of heuristics for the STSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*, pages 369–443. Kluwer Academic, Norwell, MA, 2002.

R. Jonker and T. Volgenant. Transforming asymmetric into symmetric traveling salesman problems. *Operations Research Letters*, 2:161–163, 1983.

R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

J. R. King. Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm. *International Journal of Production Research*, 18(2): 213–32, 1980.

A. Kusiak. Analysis of integer programming formulations of clustering problems. *Image and Vision Computing*, 2:35–40, 1984.

A. Kusiak. Flexible manufacturing systems: A structural approach. *Int. J. Production Res.*, 23:1057–73, 1985.

E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem*. John Wiley & Sons, Essex, England, 1985.

J.K. Lenstra. Clustering a data array and the Traveling Salesman Problem. *Operations Research*, 22(2):413–4, 1974.

J.K. Lenstra and A. H. G. Rinnooy Kan. Some simple applications of the Travelling Salesman Problem. *Operational Research Quarterly*, 26(4):717–733, 1975.

A. Lim, B. Rodrigues, and X. Zhang. Metaheuristics with local search techniques for retail shelf-space optimization. *Informs*, 50(1):117 –131, 2004.

S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

Y. Liu, B. J. Ciliax, A. Pivoshenko, J. Civera, V. Dasigi, A. Ram, R. Dingledine, and S. B. Navathe. Evaluation of a new algorithm for keyword-based functional clustering of genes. In *8th International Conf. on Research in Computational Molecular Biology (RECOMB-04)*, San Diego, CA, March 2004. Poster paper.

A. Lodi and A. P. Punnen. TSP software. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*. Kluwer Academic, Norwell, MA, 2002.

S.T. March. Techniques for structuring data base records. *Computing Surveys*, 15:45–79, 1983.

F. Marcotorchino. Block seriation problems: A unified approach. *Appl. Stochastic Models and Data Analysis*, 3:73–91, 1987.

W. T. McCormick, P. J. Schweitzer, and T. W. White. Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20:993–1009, 1972.

P. Moscato. TSPBIB. *http://www.densis.fee.unicamp .br/~moscato/TSPBIB_home.html*, web.

S. Navathe, S. Ceri, G. Wiederhold, and J. Dou. Vertical partitioning of algorithms for database design. *ACM Trans. Database Syst.*, 9(4):680–710, 1984.

M.T. Ozsu and P. Valduriez. *Principles of distributed database systems.* Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1999.

A. P. Punnen. The Traveling Salesman Problem: applications, formulations, and variations. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*. Kluwer Academic, Norwell, MA, 2002.

R. Shamir and R. Sharan. Algorithmic approaches to clustering gene expression data. In T. Jiang, T. Smith, Y. Xu, and M.Q. Zhang, editors, *Current Topics in Computational Biology*, pages 269–299. MIT Press, 2002.

P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273–97, 1998.

M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. Technical Report 00-034, University of Minnesota, 2000.

R. Torres-Velzquez and V. Estivill-Castro. Local search for hamiltonian path with applications to clustering visitation paths. *Journal of the Operational Research Society*, 55(7): 737–748, 2004.

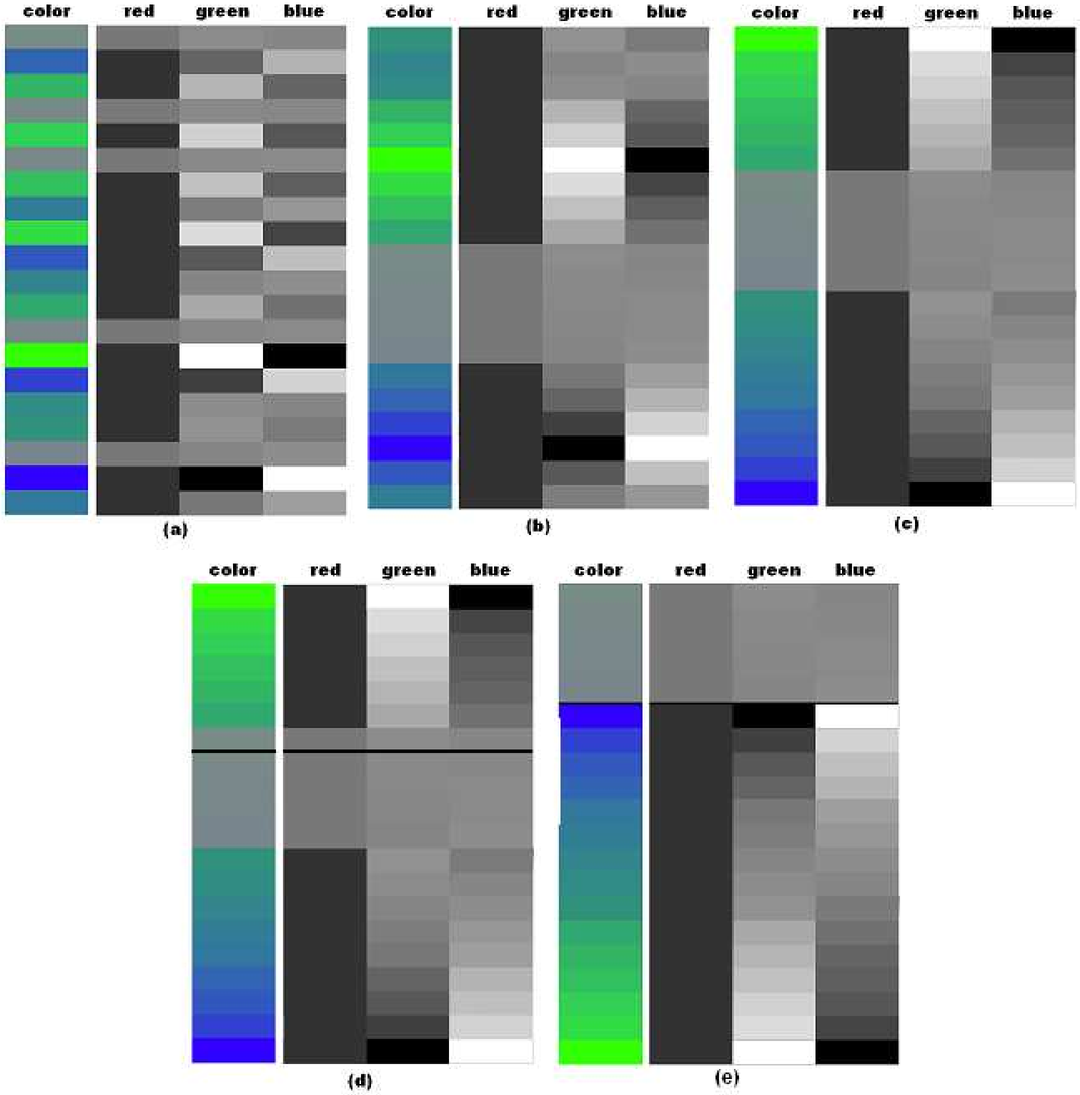

Figure 3: Rearrangement clustering of a set of color samples using their red, green, and blue components as their features. (a) The initial arrangement. (b) Rearrangement using BEA. (c) Rearrangement using TSP. This rearrangement is optimal for objective (2). (d) Restricted partitioning with $k = 2$. The black line indicates the cluster boundary. This algorithm yields the same ordering as TSP+k with $k = 1$. The gray cluster is split as the partitioning minimizes the maximum diameter. (e) Rearrangement using TSP+k with $k = 2$. The clusters are correctly identified as indicated by the black line.

Figure 4: Marketing techniques and applications example. (a) Initial matrix. (b) BEA clustering. The gray rectangles indicate the clusters identified in (McCormick et al. 1972). The black horizontal lines indicate the two non-overlapping clusters that are compared. (c) Optimal clustering with $k = 1$. (d) TSP+k solution. The horizontal lines indicate the two clusters that are compared with the BEA solution. The 4-element cluster is the same for both algorithms. The 3-element clusters differ by one item, yielding ME = 8 for BEA and ME = 10 for TSP+k.

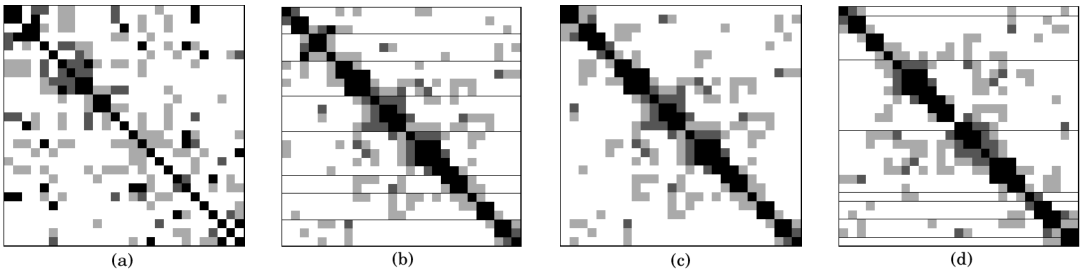

Figure 5: Airport design example. (a) Initial array. (b) BEA clustering with ME = 577 for the entire matrix. The sum of the ME values for the 8 clusters is 464. (c) Optimal clustering with $k = 1$, yielding ME = 580 for the entire matrix. (d) Optimal clustering with $k = 8$. The sum of the ME values for the 8 clusters is 503.

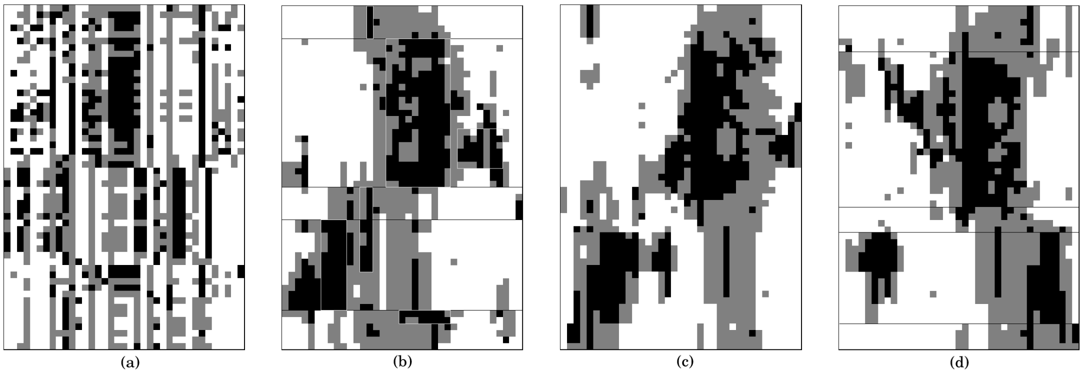

Figure 6: Aircraft types and functions example. (a) Initial array. (b) BEA clustering with ME = 1930 for the entire matrix. The sum of the ME values for the two largest clusters of aircraft is 1545. (c) Optimal clustering with $k = 1$, with ME = 1961 for the entire matrix. (d) Optimal clustering with $k = 17$. The sum of the ME values of the two largest clusters is 1616. These clusters contain the same number of aircraft as the BEA clusters with 24 and 14 aircraft respectively.

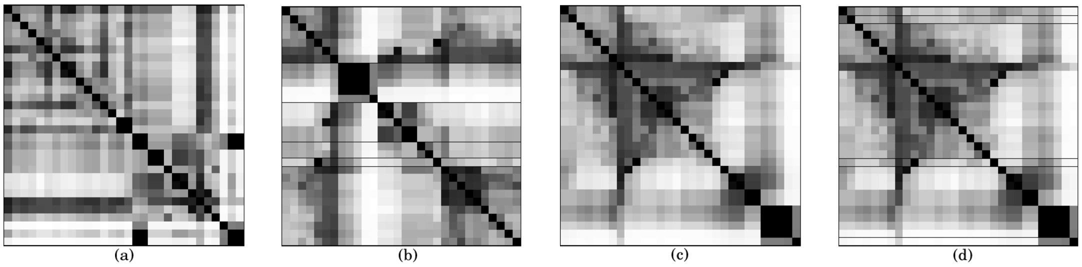

Figure 7: Personnel database records example. (a) Initial array. (b) BEA clustering with ME = 1,791,870 for the entire matrix. The sum of the ME values for the 6 clusters is 1,533,034. (c) Optimal clustering with $k = 1$ and ME = 1,836,260 for the entire matrix. (d) Optimal clustering with $k = 6$. The sum of the ME values for the 6 clusters is 1,645,207.

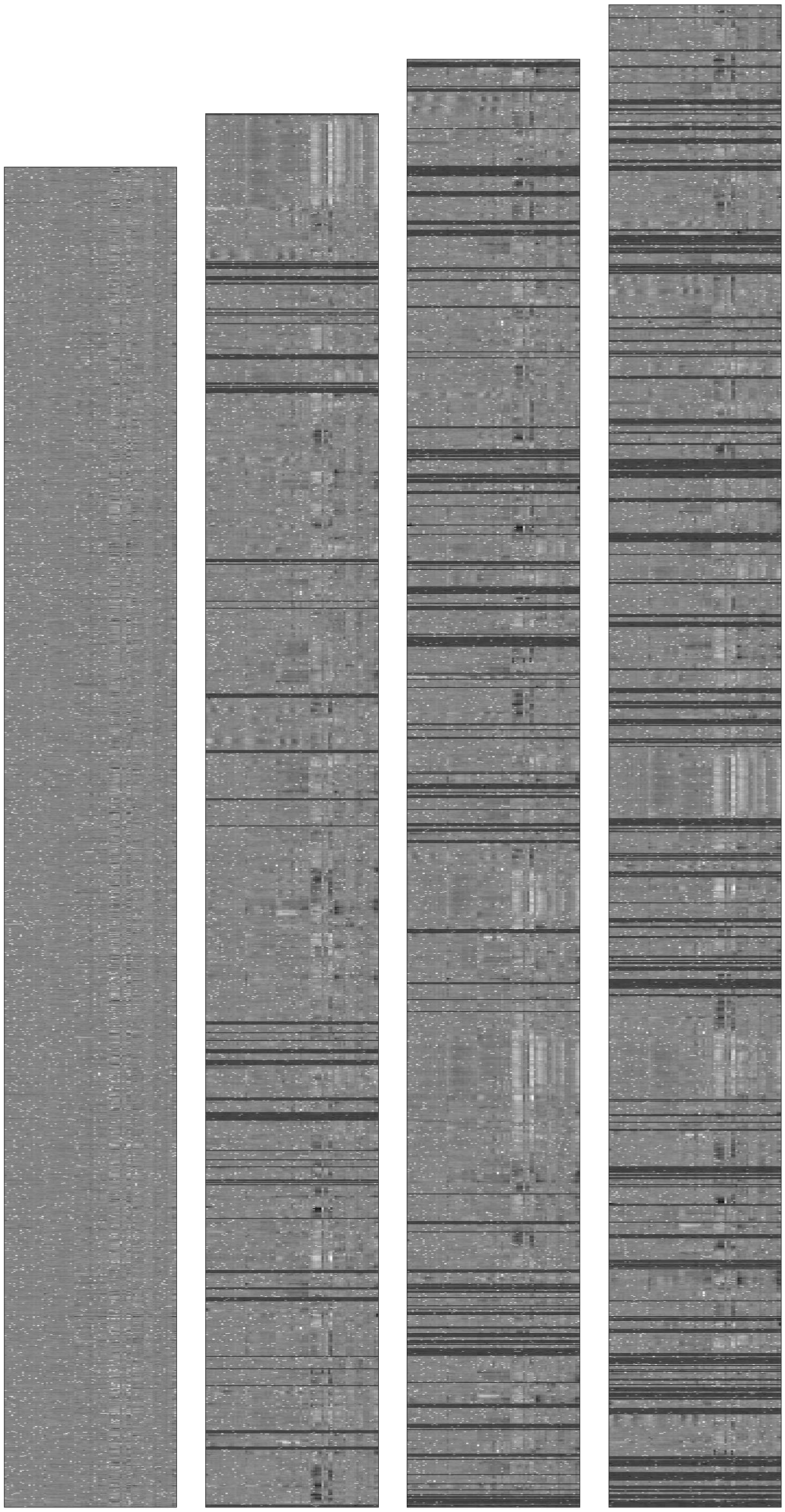

Figure 8: 2,467 yeast gene expression data randomly ordered (left) and rearranged using TSP+k with k equal to 100, 200, and 300. Cluster boundaries are marked by black lines. Missing data values are colored white.