Take a Walk and Cluster Genes: A TSP-based Approach to Optimal Rearrangement Clustering

Sharlee Climer SCLIMER@CSE.WUSTL.EDU
Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO 63130, 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, USA

Abstract

Cluster analysis is a fundamental problem and technique in many areas related to machine learning. In this paper, we consider rearrangement clustering, which is the problem of finding sets of objects that share common or similar features by arranging the rows (objects) of a matrix (specifying object features) in such a way that adjacent objects are similar to each other (based on a similarity measure of the features) so as to maximize the overall similarity. Based on formulating this problem as the Traveling Salesman Problem (TSP), we develop a new TSP-based optimal clustering algorithm called TSPCluster. We overcome a flaw that is inherent in previous approaches by relaxing restrictions on dissimilarities between clusters. Our new algorithm has three important features: finding the optimal k clusters for a given k, automatically detecting cluster borders, and ascertaining a set of most viable clustering results that make good balances among maximizing the overall similarity within clusters and dissimilarity between clusters. We apply TSPCluster to cluster and display ∼500 genes of flowering plant *Arabidopsis* which are regulated under various abiotic stress conditions. We compare TSPCluster to the bond energy algorithm and two existing clustering algorithms. Our TSPCluster code is available at (Climer & Zhang, 2004).

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

1. Introduction

Clustering, or cluster analysis, is aimed at discovering structures and patterns of a given data set. It has been studied in and applied to many research areas, such as machine learning, statistics, pattern recognition, information retrieval, data mining, computational biology and manufacturing. As a fundamental problem and technique for data analysis, clustering becomes increasingly important, especially with the explosion of data on the World Wide Web and the advent of massive quantities of genomic data.

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 & Hubert, 1990; Arabie et al., 1988). 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. Since its early studies published in the 70's, this problem has been extended to many variations and applied to many different applications in various areas, such as information retrieval (March, 1983), manufacturing (Kusiak, 1985), and software engineering (Gorla & Zhang, 1999). The core problem does not seem to have been given a consistent name; it has been referred to as "structuring of matrices" (Punnen, 2002), "data reorganization" (McCormick et al., 1972), and "clustering of data arrays" (Lenstra, 1974). The first algorithm, called *bound energy algorithm* (McCormick et al., 1972), which yields an approximate solution, has gained wide recognition. It has been pointed out correctly that the problem is equivalent to the Traveling

Salesman Problem (TSP) (Lenstra, 1974). (The TSP for n cities is the problem of finding a tour visiting all the cities and returning to the starting city minimizing the sum of the distances between consecutive cities. It is well known that TSP is NP-hard.) Due to its nature and for the convenience of our discussion, we call this clustering problem *rearrangement clustering*.

Many approximation methods have been proposed and attempted for rearrangement clustering (Arabie & Hubert, 1990). Almost all the existing methods 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.

It is generally very difficult to find the right number of clusters for a given clustering problem. One approach for gene clustering is a semi-automatic method, combining a clustering algorithm and human inspection to determine a viable number of clusters and the corresponding clusters (Eisen et al., 1998). In this approach, a clustering algorithm is first applied to arrange the genes and the result is displayed and visually inspected. This method, originally championed in (Eisen et al., 1998), is perhaps the most widely used gene clustering algorithm. Note that clustering genes for displaying and visual inspection is similar to the rearrangement clustering problem.

The previous approach to rearrangement clustering is seriously flawed when applied to data that falls into natural clusters. Consider the example illustrated in Figure (1), where objects have only two features and the dissimilarity between objects is the Euclidean distance. When these objects are arranged to minimize the overall dissimilarity, the large cluster starting at object a and ending at object d is broken in half and placed at each end of the ordering. Although objects b and c are very similar, they are separated by an entire cluster. 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 "jump" to adjacent clusters. When natural clusters occur, the sum of dissimilarities between adjacent objects is dominated by inter-cluster dissimilarities, as these values are substantially larger than intra-cluster dissimilarities. In this paper, we propose an alternative objective function that addresses this defect and present a technique for solving this new objective.

The primary motivating problem of our research is gene clustering in computational genomics. Identifying co-expressed genes is a critical and challenging problem in molecular genetics for elucidating gene

Figure 1. Example demonstrating the splitting of a cluster.

regulatory mechanism. The current microarray gene expression technology (Baldi & Hatfield, 2002; Eisen et al., 1998) is able to examine the expressions of hundreds, thousands and 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 & Hatfield, 2002; Eisen et al., 1998). Given a gene expression matrix and a similarity function, we want to rearrange the genes so that adjacent genes are similar to each other.

We apply our rearrangement clustering technique in this context. Specifically, we develop a new method for gene clustering and displaying using microarray data. Our method is based on the formulation of the problem as the Traveling Salesman Problem (TSP); we thus name our algorithm *TSPCluster*. The main novel idea behind our new algorithm is introducing dummy objects (cities) into a mapped TSP, which is then solved *optimally*. By introducing k dummy objects into the problem, we are able to find k clusters that maximize the total similarity of objects within the same clusters and at the same time allow dissimilarity between the clusters. Moreover, the borders of the clusters are automatically determined by the algorithm. We then use the inter-cluster dissimilarity measure to help detect the most viable numbers k of clusters for the gene clustering problem. Our method is usable for large gene clustering problems thanks to recent advances in the TSP research. Our code is composed of two programs.

The first converts gene data to a TSP problem, and the second rearranges the rows of the microarray data according to the TSP solution. Any TSP solver can be used. In our experiments, we used Concorde (Applegate et al., 2001), an award winning TSP solver that has successfully solved a record 15,112-city TSP instance. By using Concorde, we can routinely solve gene clustering problems of a few thousand genes to optimality. The Concorde code is publicly available at (Applegate et al., web).

We apply our new TSPCluster algorithm to cluster ~500 genes in flowering plant *Arabidopsis*, which are regulated under various abiotic stress conditions (Seki et al., 2002). We compare our results with the bond energy algorithm (BEA). We also compare TSPCluster with two existing gene clustering algorithms, Cluster and k-ary construction. Cluster (Eisen et al., 1998), a hierarchical clustering algorithm, is the first gene clustering and displaying algorithm for microarray data and has been widely used in biology and computational biology communities. The k-ary construction algorithm (Bar-Joseph et al., 2003) improves upon Cluster by combining up to k objects at once and finds an optimal arrangement of leaf nodes in the clustering tree (however, the overall solution is still an approximation).

The paper is organized as follows. We first formulate the rearrangement clustering problem as the TSP in Section 2. We develop our TSPCluster algorithm and analyze its properties in Section 3. We then apply TSPCluster to the microarray dataset and compare the results to BEA, Cluster, and k-ary in Section 4. We discuss related work in Section 5 and conclude in Section 6.

2. Problem Formulation

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, $s(i,j)$ represent a similarity measure for objects (rows) i,j, and $V(p)$ be the sum of similarities between adjacent rows for an optimal permutation $p \in P$ for the given similarity measure. Then,

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

for n objects. Conversely, given a *dissimilarity* function, $d(i,j)$ and $W(p)$ equal to the sum of dissimilarities between adjacent rows for an optimal permutation $p \in P$,

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

Given n cities and the distances between each pair of cites, the TSP is the problem of finding a minimum length complete tour visiting each city exactly once. 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.

The mapping from a rearrangement clustering problem instance to a TSP instance is straightforward (Lenstra, 1974). 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. 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. The location of the dummy city is the optimal point for breaking the TSP cycle into a path. We make the following 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 path on the TSP tour that excludes the dummy city 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 acrossed by the dummy city. That is, $d(u,v) < d(x,y)$. Then we can directly connect cities u and v, with an increase of $d(u,v)$ to the overall distance, and insert the dummy city between cities x and y, with an decrease of $d(x,y)$ to the overall distance. This will result in a net decrease 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. The second part follows directly from the first part of the Lemma coupled with the fact that the TSP tour has the smallest possible distance among all complete tours. □

An immediate collorary of Lemma 1 is that the dissimilarity (distance) between the objects on the top and the bottom rows of the final permutated matrix is larger than any of the adjacent pairs in the permutation and the permutated arrangement of the objects gives the minimum overall dissimilarity.

Note that objective (2) is an optimization problem based on a given dissimilarity function and no assumptions are made about the properties of this function. Distance functions are frequently 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.

3. The Algorithm and Properties

Rearrangement clustering, as defined by objectives (2), is aimed to find the best total linear order of all objects so that the objective function is minimized. However, we are not interested in arranging rows in such a way that the overall dissimilarity between adjacent rows is strictly minimized. Instead, we are concerned with problems in which natural clusters may exist. The dissimilarity between two adjacent rows which define the borders of two clusters may be small when the total dissimilarity of the permutation is the minimum possible. However, it is desirable for the total dissimilarity across cluster borders to be large; the larger the total inter-cluster dissimilarity, the greater the distinction between clusters. The crucial question is, how can we find cluster borders that minimize the total intra-cluster dissimilarity while tolerating large inter-cluster dissimilarity?

For our purpose, we redefine our objective as follows:

$$W(p,k) = min\left(\sum_{i=1}^{k}\sum_{j=u}^{v-1} d(j,j+1)\right) \qquad (3)$$

where

$$u = \sum_{l=0}^{i-1} m_l \qquad (4)$$

$$v = \sum_{l=0}^{i} m_l - 1 \qquad (5)$$

and m_i is the number of objects in cluster i, $m_0 = 1$, and k is the number of clusters. In other words, we desire to minimize the intra-cluster dissimilarity while disregarding the inter-cluster dissimilarity.

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 paths and form the cluster borders. After the TSP is solved, the dummy cities and their edges are removed and replaced by the edges that connect the end nodes of the paths. Each of these borderline edges divides two clusters. The lengths of the borderline edges is not of any consequence in the solution of the TSP. In this way, the TSP solution optimizes the intra-cluster dissimilarity, while allowing inter-cluster dissimilarities to assume whatever value best suits the intra-cluster dissimilarity.

Theorem 1 *When there exist k dummy cities, the total distance of the paths on the TSP tour that are separated by dummy cities 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: We assume that the edge distances input into the TSP solver are positive values. (If nonpositive values exist in the data, a sufficiently large constant can be added to each of the n original edges and the optimal tour will be the same as the optimal tour for the original problem.) We also assume that the constant that is used for the distances to dummy cities is sufficiently small. The TSP tour will contain $n - k$ edges with distances greater than zero and $2k$ edges with distances equal to zero. Since this tour is the minimum possible, it follows that the total distance of the paths is also minimized. The rest of this proof is similar to the proof of Lemma 1. □

For a typical clustering problem, there is usually a range of values for the number of clusters k that are of interest. Consider clustering the population of the United States. 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 a very large k value.

Theorem 1 guarantees the optimality of identifying k clusters for a given k, based on the objective function (3). Assuming that a range of k values is specified, determining the best value for k within this range is the next critical problem to address.

Theorem 2 *Let*

$$d_{mean} = \frac{\sum_{i=1}^{k}\sum_{j=u}^{v-1} d(j,j+1)}{(n-k)} \qquad (6)$$

where u and v are defined as in (4) and (5). As k increases, d_{mean} is non-increasing.

Proof: d_{mean} is the average intra-cluster dissimilarity. In general, as k increases, the membership of clusters may rearrange 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 zero length edges and the new dummy node. From Theorem 1, we know that the deleted edge must have the maximum distance. Thus, the average distance of the edges cannot increase with this addition. Since the TSP finds the minimum tour distance, this property holds when the tour undergoes more than just one minor change. Therefore, the average dissimilarity within paths is non-increasing. □

The TSPCluster algorithm guarantees an optimal rearrangement clustering for a given k, however, as shown by Theorem 2, we must consider desirable qualities other than average intra-cluster dissimilarity 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 have large dissimilarities between clusters. We use this latter approach in the analysis of gene data in Section 4.

4. Applications and Comparison

We applied our new algorithm, TSPCluster, to cluster a microarray dataset containing 499 up regulated genes of flowering plant *Arabidopsis* under five evironmental conditions including drought, cold and high-salinity (Seki et al., 2002). Note that the genes in this set were identified as stress inducible genes from a larger set of 7000 genes; most of them form clusters corresponding to different function categories. Our main goal on this set of genes is to refine the clustering results so as to help identify the transcription factor binding motifs in the upstream regulatory regions of clustered genes. The data was collected in five different time points for each of the five stress conditions, resulting in a total of twenty five features for each gene.

For our comparison with the bond energy algorithm (BEA), we used the similarity measure that is employed by BEA as follows:

$$s(i,j) = \sum_{k=1}^{n} a_{ik} a_{jk} \tag{7}$$

where a_{ik} is the kth feature of object i. After finding the similarities, we apply an additive inverse to translate the values to dissimilarities for TSPCluster.

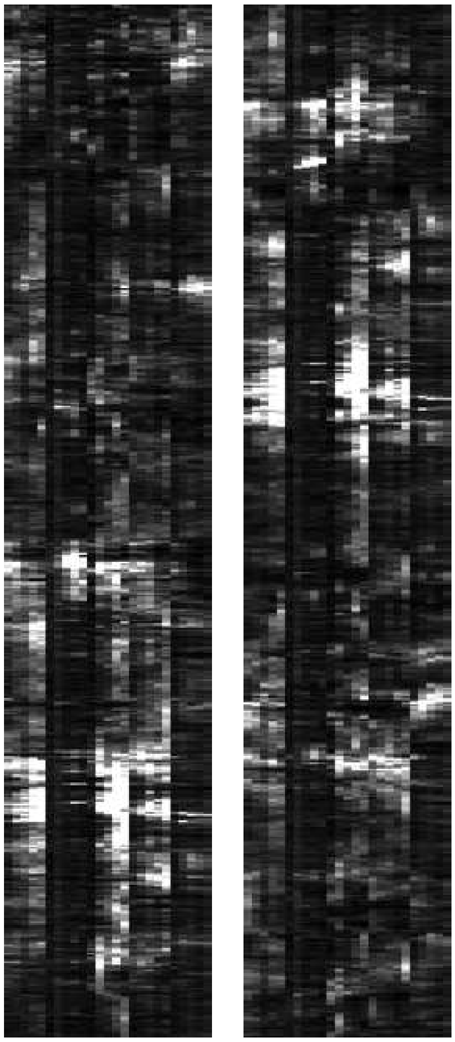

Figure 2. Gene expression data for the rearranged *Arabidopsis* dataset using (a) BEA and (b) TSPCluster algorithms and the BEA similarity measure.

We first compare BEA to TSPCluster with $k=1$ using the original objective function (1). The overall similarity score is 447,070 for BEA and 452,109 for TSPCluster. As would be expected, TSPCluster has a higher similarity score as it provides an optimal solution. In Figure (2), we display the gene expression levels rearranged by the two algorithms. We then apply TSPCluster with k equal to two through fifty and observe that average inter-cluster dissimilarities have local peaks at k equal to 6, 13, 19, 26, 29, 35, 40, and 47.

The computation time varied for these TSPCluster trials. Some instances took less than a half minute while others took about three minutes to solve. (We ran these trials on an Athlon 1.9 MHz dual processor with two gigabytes shared memory.)

For our comparisons with Cluster (Eisen et al., 1998) and k-ary construction (Bar-Joseph et al., 2003), we used the Pearson correlation coefficient for our similarity measure as follows:

$$s(x,y) = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{\left(\sum X^2 - \frac{(\sum X)^2}{N}\right)\left(\sum Y^2 - \frac{(\sum Y)^2}{N}\right)}} \quad (8)$$

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. This similarity function was also used for comparisons of algorithms for clustering gene expression data in (Shamir & Sharan, 2002).

We first compare these implementations with TSPCluster with $k = 1$ using the objective function (1). $W(p)$ is equal to 398, 427, and 436 for Cluster, k-ary, and TSPCluster, respectively. Figure (3) shows the graphical display of these clusterings for the *Arabidopsis* data set.

We ran TSPCluster for k equal to one through fifty and observed that the average inter-cluster dissimilarities have local peaks at k equal to 3, 9, 12, 21, 26, and 40. Thus, k equal to 26 and 40 yielded local maxima for both similarity functions. Figure (4) shows the gene expression levels, using TSPCluster with k set to 26 and 40.

In our experiments, we noticed a substantial number of singletons were found. These genes are not of interest and must be quite dissimilar to the rest of the data to appear as singletons when the value of k is relatively small. Furthermore, their large dissimilarity may skew results when they are included in a cluster. For these reasons, we plan to update our code to signal the existence of singletons so we can eliminate them from the data set.

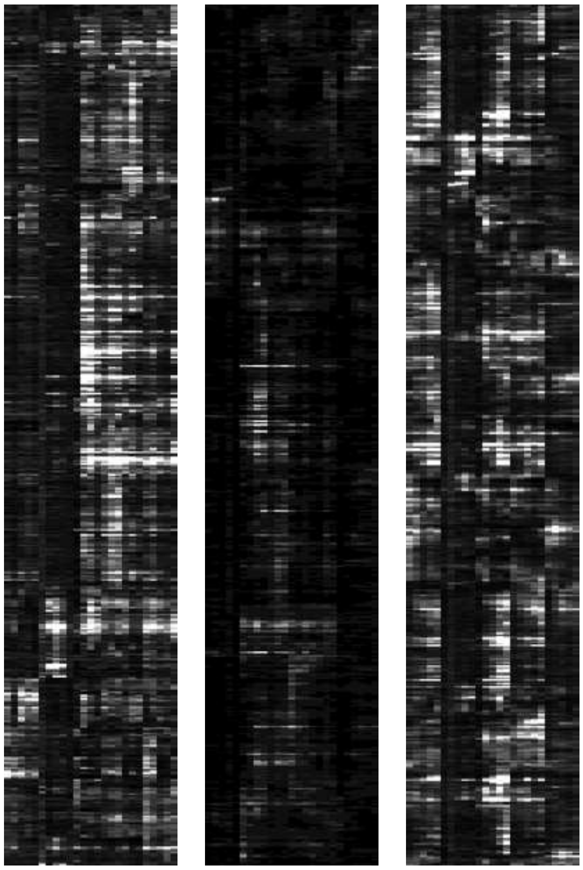

Figure 3. Gene expression data for the *Arabidopsis* data set rearranged using (a) Cluster, (b) k-ary, and (c) TSPCluster algorithms.

5. Related Work and Discussions

Similar to other clustering problems, gene clustering is difficult and often intractable in the worst case. Therefore, many heuristic and approximation methods have been proposed and developed. The most popular and extensively applied gene clustering algorithm for microarray data is a hierarchical clustering algorithm called Cluster (Eisen et al., 1998). Hierarchical clustering is a simple approximation method. It progressively combines two genes or objects that have the closest similarity into one object with the orginal genes or objects removed from further consideration. The process continuously builds a tree of objects until it produces one single object.

The hierarchical clustering algorithm has been recently extended to allow up to k objects (as opposed to just two objects) to be combined at a time, followed by a rearrangement of the leaf nodes of the clustering tree to improve the overal similarity score, resulting in an extended hierarchical algorithm called k-ary clustering (Bar-Joseph et al., 2003). In essence, hierarchical clustering is a greedy bottom up approach. In addition to being unable to guarantee optimality of the final result, it also suffers from the problems of not providing information of how many clusters the underlying data set is most likely to have and where the borders of clusters lie.

The other popular approach is the partition based method. This approach is represented by the well-known k-means algorithm, which attempts to find the best possible k clusters for a given k. The partitioning approach yields the borders of the clusters for a given k, however, in addition to being an approximation method, it is not suitable for rearrangement clustering as it fails to determine an ordering within the clusters.

The TSPCluster method yields the optimal ordering for minimizing intra-cluster dissimilarities. Furthermore, it tolerates inter-cluster dissimilarity and automatically defines cluster borders.

In some applications, there may exist a dissimilarity function that is not strictly symmetric. That is, $d(i, j)$ may not necessarily be equal to $d(j, i)$. In this case, the TSPCluster algorithm is still viable. Instead of solving a symmetric TSP (STSP), an asymmetric TSP (ATSP) would be computed. Concorde is designed for STSP instances, however, ATSP instances can be converted to STSP instances using a *2-node* transformation (Jonker & Volgenant, 1983), in which the number of cities is doubled.

6. Conclusion

In this paper, we present an algorithm, TSPCluster, that solves the rearrangement clustering problem optimally for a given number of clusters k. By modeling the problem as a TSP, this algorithm minimizes the dissimilarity of adjacent rows within each cluster and ignores the dissimilarity of adjacent rows that define the borders of two clusters. Furthermore, the algorithm returns the location of cluster borders, providing a solution to a previously difficult problem for rearrangement clustering. The algorithm can be run once for each feasible value of k and the quality of the results can be compared. In this paper, we used the dissimilarity between clusters that resulted from the rearrangement to determine which of these results may be a viable arrangements for consideration.

While we have focused on finding optimal solutions in this paper, many applications require very fast computation times and optimality is not feasible. Fortunately, there has been a vast amount of research devoted to quickly finding approximate solutions to the TSP, yielding a wealth of available code (Lodi & Punnen, 2002; Moscato, web).

In the future, we plan to modify our code to order the clusters that are returned after solving the TSP. Since the inter-cluster values are not considered in the solution of the TSP, the clusters produced could take on any arbitrary order. We plan to reapply the TSP to these clusters and identify the optimal inter-cluster dissimilarities.

Our TSPCluster code is available at (Climer & Zhang, 2004).

Acknowledgments

This research was supported in part by NDSEG and Olin Fellowships and by NSF grants IIS-0196057 and ITR/EIA-0113618. We are grateful to the anonymous reviewers for identifying weak areas of the paper and providing valuable insights.

References

Applegate, D., Bixby, R., Chvatal, V., & Cook, W. (2001). TSP cuts which do not conform to the template paradigm. In M. Junger and D. Naddef (Eds.), *Computational combinatorial optimization*, 261–304. Springer.

Applegate, D., Bixby, R., Chvátal, V., & Cook, W. (web). Concorde - A code for solving Traveling Salesman Problems. 15/12/99 Release, *http:www.keck.caam.rice.edu/concorde.html*.

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

Arabie, P., Schleutermann, S., Daws, J., & Hubert, L. (1988). Marketing applications of sequencing and partitioning of nonsymmetric and/or two-mode matrices. In W. Gaul and M. Schader (Eds.), *Data analysis, decision support, and expert knowledge representation in marketing*, 215–24. Springer Verlag.

Baldi, P., & Hatfield, G. (2002). *Dna microarrays and gene expression*. Cambridge University Press.

Bar-Joseph, Z., Demaine, E., Gifford, D., Hamel, A., Jaakkola, T., & Srebro, N. (2003). K-ary clustering with optimal leaf ordering for gene expression data. *Bioinformatics*, *19*, 1070–8.

Climer, S., & Zhang, W. (2004). Rearrangement clustering code. *http://www.climer.us* or *http://www.cse.wustl.edu/~zhang/projects/software*.

Eisen, M., Spellman, P., Brown, P., & Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy Sciences*, *95*, 14863–8.

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

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

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

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

Lodi, A., & Punnen, A. P. (2002). TSP software. In G. Gutin and A. Punnen (Eds.), *The traveling salesman problem and its variations*. Norwell, MA: Kluwer Academic.

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

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

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

Punnen, A. P. (2002). The Traveling Salesman Problem: applications, formulations, and variations. In G. Gutin and A. Punnen (Eds.), *The traveling salesman problem and its variations*. Norwell, MA: Kluwer Academic.

Seki, M., Narusaka, M., Ishida, J., Nanjo, T., Oono, Y. F. M., Kamiya, A., Nakajima, M., Enju, A., Sakurai, T., Satou, M., Akiyama, K., Taji, T., Yamaguchi-Shinozaki, K., Carninci, P., Kawai, J., Hayashizaki, Y., & Shinozaki, K. (2002). Monitoring the expression profiles of 7000 arabidopsis genes under drought, cold and high-salinity stresses using a full-length cdna microarray. *Plant Journal*, *31*, 279–92.

Shamir, R., & Sharan, R. (2002). Algorithmic approaches to clustering gene expression data. *Current Topics in Computational Biology* (pp. 269–299). MIT Press.

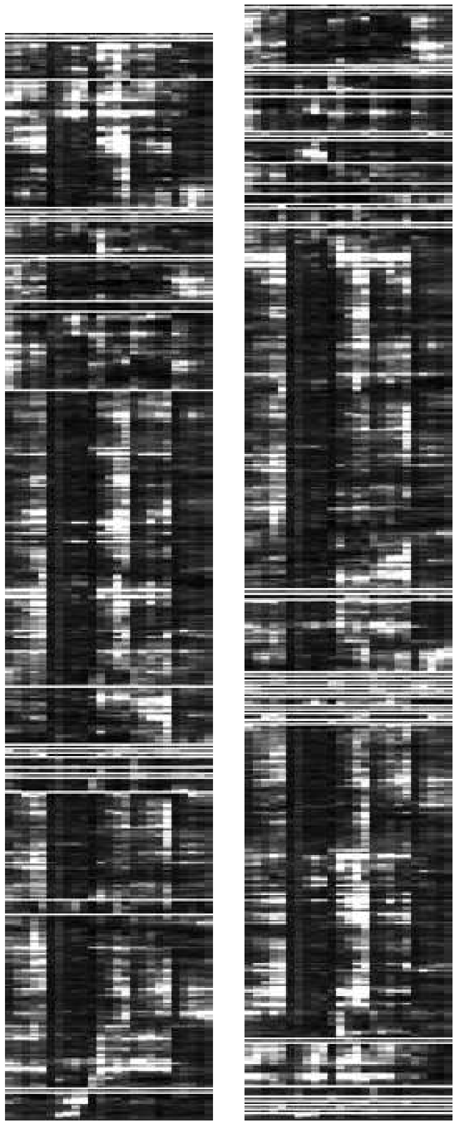

Figure 4. Gene expression data for the *Arabidopsis* data set rearranged using the TSPCluster algorithm. (a) $k = 26$ (b) $k = 40$. The solid white lines are cluster borders.