Partition-Distance: A Problem and Class of Perfect Graphs Arising in Clustering

In Information Processing Letters, Volume 82, Issue 3, 16 May 2002, Pages 159-164

Dan Gusfield[1]
Department of Computer Science, University of California, Davis

Abstract

Partitioning of a set of elements into disjoint clusters is a fundamental problem that arises in many applications. Different methods produce different partitions, so it is useful to have a measure of the similarity, or distance, between two or more partitions. In this paper we examine one distance measure used in a clustering application in computational genetics. We show how to efficiently compute the distance, and how this defines a new class of perfect graphs.

Keywords: partitioning, clustering, assignment problem, node cover, perfect graph, genetics, graph algorithms, combinatorial problems

[1]Research partially supported by grant DBI-9723346 from the National Science Foundation. email: gusfield@cs.ucdavis.edu

1 Introduction

The problem and the graphs discussed in this paper arise from a problem considered in [1], where an algorithm is developed for deducing the family structure of a set of individuals (fish actually) given particular genetic data (the details are not central here). The output of the algorithm is a partition of the individuals into some number of clusters. The true family structure is also a partition of the individuals, possibly the same partition.

In order to evaluate the goodness of a partition produced by the algorithm, in cases where the true family structure is known from expensive laboratory work or from simulations, a *distance* between two partitions was defined in [1]. Then, an *exponential-time* algorithm (worst-case, as a function of the set size) was presented to compute that distance, given any two partitions. The issue of computing a distance (or similarity) between two partitions is a natural one that arises in many other clustering applications. For example, clustering of sequence and molecular expression data has become a huge concern in computational biology, but many different clustering methods are employed, and different partitions of the same data are produced. We need a way to measure the closeness of those partitions, and even cluster the partitions based on that measure.

In this paper, we first show how to compute the distance between two partitions in polynomial time. From a practical standpoint, that is the main point of interest. However, the problem of computing the distance between two (or more) partitions has a natural interpretation as a *node-cover problem* on a graph derived from the partitions. Since the distance problem for two partitions has a polynomial-time solution (using the details of the clusters), while the node-cover problem is NP-hard in general (given an arbitrary graph as input), we were interested in characterizing the graphs that are *derived* from the partition-distance problem, to determine if the node-cover problem can be solved in polynomial time on these graphs (where the original partition details are not part of the input). We characterize the structure of those graphs and show that they form a class of *perfect* graphs, implying that the node-cover problem on those graphs can be solved in polynomial time [6]. We also give a polynomial-time algorithm to determine if a graph is in this class, and to create two partitions that would generate the graph. Finally, we note that the partition-distance problem is NP-hard for three partitions.

Definitions: Given a set of elements N, a *cluster* is a non-empty subset of N. A *partition* of N is a set of mutually exclusive clusters whose union is N. The number of clusters is limited only by $|N|$. Two partitions P and P' of N are *identical* if and only if every cluster in P is a cluster in P' (the converse is then forced). Given two partitions P and P' of N, the *partition-distance,* $D(P, P')$, between P and P' is the *minimum* number of elements that must be deleted from N, so that the two induced partitions (P and P' restricted to the remaining elements) are identical. (See Table 1).

The partition-distance is also equal to the minimum number of elements that must be *moved* between clusters in P, so that the resulting partition equals P' (by definition, any set that becomes empty is no longer a cluster). This alternative definition may be more meaningful in some applications. The fact that the two definitions give the same distance is left to the reader.

2 Efficient Solution

Theorem 2.1 *The partition-distance can be computed in polynomial time.*

Proof An instance of the classical *As-*

	S_1	S_2	S'_1	S'_2	S'_3
1	*1*	*0*	*1*	*0*	*0*
2	*1*	*0*	*1*	*0*	*0*
3	*0*	*1*	*0*	*0*	*1*
4	*1*	*0*	*0*	*1*	*0*
5	*0*	*1*	*0*	*1*	*0*
6	*1*	*0*	*1*	*0*	*0*

Table 1: Partition P consists of clusters S_1 and S_2. Partition P' consists of S'_1, S'_2 and S'_3. A value of 1 indicates membership in the cluster. $D(P, P') = 2$: after the removal of elements 3 and 4, S_1 and S'_1 become identical, S_2 and S'_2 become identical, and S'_3 becomes empty and hence is no longer a cluster.

signment Problem [7, 3] consists of a matrix of numbers M, and an *assignment* is a selection of cells of M such that no row or column contains more than one selected cell. An *optimal assignment* is an assignment whose selected cell values have the largest sum over all possible assignments. An optimal assignment can be computed in polynomial time as a function of the size of M.

To solve the partition-distance problem, create an instance $M(P, P')$ of the assignment problem with one row i for each cluster S_i in P and one column j for each cluster S'_j in P'. Associate cell (S_i, S'_j) with the subset $S_i \cap S'_j$ and write the number $|S_i \cap S'_j|$ in cell (i, j). Next, solve the assignment problem on $M(P, P')$ and let $A(P, P')$ denote the value of the assignment. We claim that $D(P, P') = |N| - A(P, P')$. Moreover, the elements to remove from N are all those elements not associated with any selected cells of the optimal assignment.

We now establish correctness. The selected cells in an assignment specify a one-one mapping between some clusters in P and an equal number of clusters in P'. After removing the elements of N not associated with selected cells, any two clusters (S_i and S'_j say) that are paired by the assignment contain the same elements ($S_i \cap S'_j$), so the two induced partitions are identical. Hence, $D(P, P') \leq |N| - A(P, P')$. Conversely, consider a selection of $D(P, P')$ elements of N whose removal causes the two induced partitions to be identical. Call that partition P''. By definition, each cluster $Z \in P''$ is a subset of some cluster S_i in P and a subset of some cluster S'_j in P', so $|Z| \leq |S_i \cap S'_j|$. Also by definition, no elements in S_i or S'_j can be in any other cluster of P''. Therefore P'' defines a one- one mapping between clusters of P and clusters of P', and hence an assignment in $M(P, P')$. It follows that $|N| - D(P, P') \leq A(P, P')$, and so $D(P, P') \geq |N| - A(P, P')$. □

Matrix $M(P, P')$ can be created in $O(|N|)$ time with the appropriate data structure. Thereafter, the classical running time for the assignment problem is $O(s^3)$ [7, 3], where s is the sum of the number of *clusters* of P and the number of clusters of P', no matter how large N is.

3 A Node-Cover Interpretation

The problem of computing the partition-distance can be cast naturally as a *node-cover* problem on a graph derived from the partitions. With this view, the exponential-time method in [1] is the standard branching algorithm for the node-cover problem. In this section we explore the node-cover approach to the partition-distance problem, connecting it to the field of perfect graphs.

Definitions: Given P and P', define the graph $G(P, P')$ with one node for each element in N; name each node by the associated element. Connect two nodes x and y by an undirected edge if and only if x and y are together in one cluster of either P or P',

but in a different cluster in the other partition. $G(P, P')$ is called a *partition graph*. A *node-cover* of a graph is a set of nodes Q such that every edge in the graph is incident with at least one node in Q. Let $N(G)$ denote the size of the smallest node cover of a graph G. The problem of computing a node-cover of minimum size is well-known to be NP- hard [4].

Lemma 3.1 *For any pair of partitions P, P' of the same set, $D(P, P') = N(G(P, P'))$.*

Proof Clearly, if elements x and y are together in one cluster of P (or P') but not together in any cluster of P' (or P), then at least one of those elements must be removed in order to create identical induced partitions. Hence the removed elements form a node-cover of $G(P, P')$. Conversely, consider a node-cover C of $G(P, P')$, remove the associated elements from N, and suppose that the induced partitions are not identical. That is, for some remaining element x, the induced cluster of P containing x and the induced cluster of P' containing x are not equal sets. Hence there is some element y contained in exactly one of those two induced clusters. But each induced cluster is a subset of a cluster in P or P' respectively, so y must be together in exactly one cluster with x in P and P', and hence (x, y) is an edge in $G(P, P')$. Therefore, if both x and y remain, C could not be a node-cover of $G(P, P')$, and hence any node-cover of $G(P, P')$ defines a set of elements of N whose removal causes the two induced partitions to be identical. □

Since the node-cover problem is NP-hard in general, but the partition-distance can be computed in polynomial time, it is natural to ask if the node-cover problem can be computed efficiently when restricted to partition graphs. We next show that the answer is yes, by first characterizing the structure of partition graphs.

3.1 The structure of partition graphs

Let $S_1, S_2, ..., S_f$ be the clusters of partition P, and let $S'_1, S'_2, ..., S'_{f'}$ be the clusters of P'. Figure 1 displays two partitions P and P' of set N in a rectangular "array". One "row" is allocated to each cluster of partition P, and one "column" is allocated to each cluster of partition P'. Each "cell" (i, j) in the array contains the elements of $S_i \cap S'_j$. Since each element is in a unique cluster of P and a unique cluster of P', each element is in a unique cell of the array.

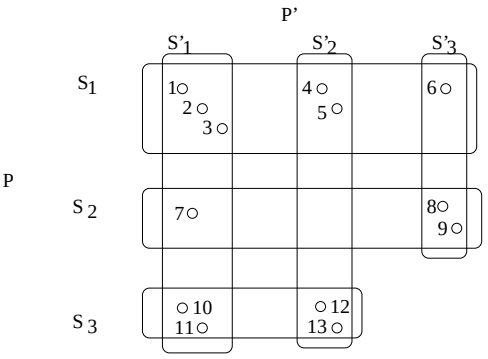

Figure 1: Partitions P and P' in arrayed layout.

We finish the construction of $G(P, P')$ by turning each element into a node and adding the edges. There is no edge between any pair of nodes that are in the same cell, since they are together in the same cluster of P and of P'. However, every node in a cell (i, j) of $G(P, P)$ is adjacent to every node in row i or column j outside of that cell (see Figure 2). These two types of edges (row and column edges) are the only edges in G, for any other edge would connect two nodes corresponding to two elements which are in different clusters of P and also in different clusters of P', contradicting the construction rules for $G(P, P')$. Hence, every row of a partition graph is a complete *multi-partite* graph. By symmetry, this is also true of each column. This layout of $G(P, P')$ is called an *arrayed layout*. Note that although every partition

graph $G(P, P')$ has an arrayed layout, since P and P' are not given as input, the arrayed layout cannot be constructed as in the above existence proof.

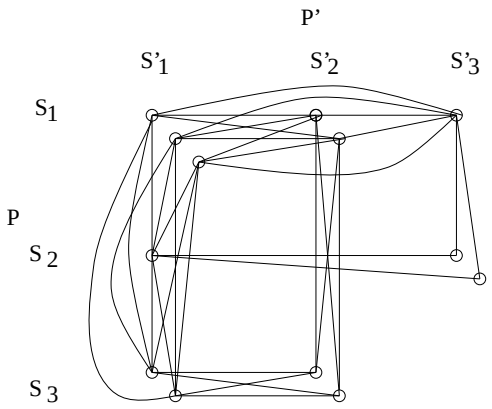

Figure 2: Partition-distance graph $G(P, P')$ in arrayed layout.

Now, suppose a graph G has an arrayed layout, i.e. G can be laid out into "rows" and "columns" so that each row is a complete multi-partite graph, as is each column, and each level in a multi-partite graph consists of the nodes of one cell of the layout. Then $G = G(P, P')$ for some partitions P and P' of a set N constructed as follows: N consists of one element for each node in G; each set S_i of partition P consists of the elements from row i; and each set S'_j of partition P' consists the elements from column j. Hence,

Theorem 3.1 *A graph G is a partition graph if and only if it has an arrayed layout.*

3.2 Efficient construction of an arrayed layout

Given a partition graph $G(P, P')$ without knowing P and P', how can an arrayed-layout of $G(P, P')$ be efficiently constructed? In an arrayed layout of G, any two nodes in any single cell of the layout have the same set of neighbors in G. Conversely, consider an arrayed layout of G, and suppose the set of nodes neighboring x in G is identical to the set of nodes neighboring y in G, but x and y are in different cells of the layout. Then moving x to the cell containing y again results in an arrayed layout of G. Hence, if there is an arrayed layout of G, there is one where two nodes of G are in the same cell of the layout if and only if they have the same set of neighbors in G.

Using the above observation, the first step in creating an arrayed layout of G is to sort the rows of the adjacency matrix of G, to form blocks of identical rows. Each such block identifies the nodes of a single cell of the layout. Considering each row as a binary number, the sorting can be done in $O(n^2)$ time by radix sort, where G has n nodes. Since each node in a cell has the same set of neighbors, we can ignore all but one representative node in each cell, and first find an arrayed layout of these. This simplifies the exposition, and we assume that each cell has only a single node.

Permuting the rows and columns of an arrayed layout still results in an arrayed layout, so we can arbitrarily choose any node v to be in the "upper left" cell. The other nodes in the first row and column of the layout are neighbors of node v. To divide those neighbors into row nodes and column nodes, let u be any neighbor of v in G, and find the set I of nodes that are neighbors of both u and v. Then the first row of the layout can be v followed by u, followed by the nodes in I in any order. The first column of the layout can be v followed by the neighbors of v not in $\{I \cup u\}$. This part of the layout can be determined in $O(n)$ time.

To finish the layout, successively consider each unplaced node w and determine the row and column to place w. To determine the row, pick a node u in each non-empty row i and check if u is adjacent to w in G. If there is such a row i, then w goes into row i; otherwise w goes into a new row. Do a simi-

lar computation to determine which existing column w goes into, or to place w in a new column. The time to place a single new node is $O(n)$, so $O(n^2)$ overall. Finally, add in the duplicate nodes for each cell, duplicate the required edges, and check that the resulting (node labeled) graph is G. In summary,

Theorem 3.2 *In $O(n^2)$ time, we can determine if a graph G has an arrayed layout, and construct one, if there is one.*

Theorem 3.3 *If G is a partition graph created from partitions P and P', then the optimal node cover of G can be found in polynomial time, even if P and P' are not known.*

Proof By Theorems 3.1 and 3.2, if G is a partition graph, then it has an arrayed layout which can be constructed in polynomial time. From that layout, we can construct two partitions, Q and Q' of set N, so that $G = G(Q, Q')$, and use those to set up and solve the assignment problem on matrix $M(Q, Q')$. By Lemma 3.1 and Theorem 2.1, an optimal node cover of G is determined by the optimal assignment in $M(Q, Q')$. Note that since the arrayed layout of G is not unique, the partitions Q and Q' are not unique either, but the value of the optimal assignment will be the same for any such pair of partitions, since the optimal node cover of G is not affected by the specific layout chosen. □

To actually solve the node-cover problem on G, we short-circuit the procedure suggested above: Once the representative nodes are placed in an arrayed layout of G, construct the matrix M by setting $M(i, j)$ equal to the full number of nodes that would go in cell (i, j) of the layout, and solve the assignment problem on M.

3.3 Perfect Graphs

Definitions: A *clique* in a graph is a subset of nodes which are pairwise adjacent; let $K(G)$ be the size of the largest clique in graph G. An *independent set* of nodes is a set of nodes where no two of the nodes are adjacent; let $I(G)$ be the size of the largest independent set in graph G. A *clique cover* of a graph is a node-disjoint set of cliques that contain every node in the graph; let $C(G)$ be the number of cliques in the clique cover of G with the fewest cliques. Clearly, $K(G) \geq I(G)$ for any graph G. A graph G is called *perfect* if $K(G') = I(G')$ for every *vertex-induced* subgraph G' of G.

See [5, 2] for a general introduction to perfect graphs. Perfect graphs are of interest because it is known that several NP-hard problems, node cover, clique cover, maximum clique, minimum coloring, and independent set, can all be solved in polynomial-time when restricted to perfect graphs [6, 2, 3]. We have established that node cover can be solved in polynomial time on partition graphs, but that is not a sufficient condition for a graph to be perfect.

Theorem 3.4 *Any partition graph is perfect.*

Proof Let $G(P, P')$ be a partition graph created from partitions P and P'. The subgraph of $G(P, P')$ induced by a subset of nodes S is just the partition graph created from the partitions P and P' restricted to S. Therefore every vertex-induced subgraph of a partition graph is itself a partition graph, and we only need to show that $I(G(P, P')) = C(G(P, P')$ for any partition graph $G(P, P')$.

Let M be a matrix of non-negative integers, and let $A(M)$ denote the value of the optimal assignment in M. It is a classic result in matching theory [7] (following from linear programming duality), that non- negative integers $R(i)$ and $C(j)$ can be found for each row i and each column j, respectively of M, so that a) $\sum_i R(i) + \sum_j C(j) = A(M)$, and b) $R(i) + C(j) \geq M(i, j)$, for any cell

(i, j) in M. Call this the *assignment duality theorem.*

Consider an arrayed layout of $G(P, P')$. We have seen above that $I(G(P, P)) = A(P, P')$, the optimal assignment value in matrix $M(P, P')$. To find a clique-cover of equal size, consider the subgraph induced by selecting exactly one node from each non-empty cell in a row i of the layout. Since each row is a complete multi-partite graph, that subgraph is a clique. Hence if we form $R(i)$ such cliques in each row i, and $C(j)$ such cliques in each column j of the layout, cell (i, j) will have nodes in $R(i) + C(j) \geq M(i, j)$ cliques. By assigning each node in each cell (i, j) to at least one of these $R(i) + C(j)$ cliques, we see that these cliques form a clique cover of $G(P, P')$. By the assignment duality theorem, this clique cover has exactly $A(P, P')$ cliques, and since $A(P, P') = I(G(P, P'))$, the theorem is proved. □

4 Generalizations

Definition: Given k partitions of N, the distance between (or similarity of) these partitions is defined as the minimum number of elements of N to remove so that all the k induced partitions are identical.

Let G be a graph on $|N|$ nodes associated with the elements of N. Two nodes are adjacent if and only if the two nodes are together in the same cluster in at least one partition, but not together in the same cluster in all k partitions. By essentially the same proof of Lemma 3.1, the distance between the partitions is given by the optimal node cover of G. However, in the case of three or more partitions, the problem of computing that distance is NP-hard (an easy reduction from 3-D matching [4]).

Another natural generalization is to attach a weight to each element v in N reflecting the reliability of the information concerning v. The *weighted distance* between two partitions P and P' of N is then the smallest *sum* of weights of any subset of N whose removal causes the two induced partitions to be identical. The problem of computing the weighted distance is also cast as an assignment problem, by setting $M(i, j)$ to the total weight of the elements in $S_i \cap S'_j$.

5 Acknowledgement

I would like to thank Jennifer Beyer for bringing the paper [1] to my attention, and to the referees for their thoughtful comments.

References

[1] A. Almudevar and C. Field. Estimation of single generation sibling relationships based on DNA markers. *J. Agricultural, biological and environmental statistics*, 4:136–165, 1999.

[2] A. Brandstadt, V.B. Le, and J.P. Spinrad. *Graph Classes - A survey.* SIAM Monographs on Discrete Math and Applications, 1999.

[3] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver. *Combinatorial Optimization.* Wiley-Interscience Publications, 1998.

[4] M. Garey and D. Johnson. *Computers and intractability.* Freeman, San Francisco, 1979.

[5] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980.

[6] M. Grotschel, L. Lovasz, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, pages 169–197, 1981.

[7] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York, USA, 1976.