
Journal of Combinatorial Optimization, 10, 77–92, 2005

Clustering of Microarray data via Clique Partitioning

GARY KOCHENBERGER Gary.Kochenberger@Cudenver.edu
School of Business, University of Colorado at Denver

FRED GLOVER Fred.Glover@Colorado.edu
Leeds School of Business, University of Colorado at Boulder

BAHRAM ALIDAEE balidaee@bus.olemiss.edu
School of Business, University of Mississippi

HAIBO WANG hwang@tamiu.edu
School of Business, Texas A&M International University

Abstract. Microarrays are repositories of gene expression data that hold tremendous potential for new understanding, leading to advances in functional genomics and molecular biology. Cluster analysis (CA) is an early step in the exploration of such data that is useful for purposes of data reduction, exposing hidden patterns, and the generation of hypotheses regarding the relationship between genes and phenotypes. In this paper we present a new model for the clique partitioning problem and illustrate how it can be used to perform cluster analysis in this setting.

Keywords: clustering, clique partitioning, metaheuristics

1. Introduction

The development of Microarrays has created new application opportunities for data mining methodologies. The potential to add to our understanding of the genome via such exploration is creating new interest in existing data mining methods. Moreover, it is motivating research into the development and testing of new models and solution approaches intended to enhance the performance of traditional methods. This array of data mining methodologies offers great potential for advances in functional genomics and molecular biology.

An early step in the mining of Microarray data often involves the clustering the data into similar groups with the intention of exposing hidden patterns of gene expression as well as suggesting possible hypotheses to be tested by other means regarding the relationship between genes and phenotypes. Clustering also serves as an effective tool for data reduction.

Much of the application of cluster analysis on Microarray data is conducted by applying one of the standard methods adopted from the statistics literature, such as K-means, Hierarchical methods, Self Organizing Maps, or some variation of these basic approaches. Excellent recent surveys of the methods and application of CA for mining Microarray data are given by Jiang et al. (2004), and Shannon et al. (2003). In recent years, however,

developments coming from metaheuristics, along with new modeling constructs, have contributed new methods with potential application to clustering problems. Models designed for clique partitioning along with their proposed solution methodologies are illustrative of these advances. The purpose of this paper is to present a new model for clique partitioning and to show its potential application to clustering of Microarray data.

In the sections below we first present the classic model for clique partitioning followed by our new model. Computational experience is given showing a comparison of the two models. We then address the application of clique partitioning to clustering of Microarray data followed by some computational experience with data sets from St. Jude Children's Research Hospital in Memphis, Tennessee. This is followed by our summary and conclusions.

2. Clique partitioning model

Consider a graph $G = (V, E)$ with n vertices and unrestricted edge weights. The clique partitioning problem (CP) consists of partitioning the graph into cliques, i.e., complete sub-graphs where every pair of nodes is connected by an edge, such that the sum of the edges weights over all cliques formed is as large as possible. This is an NP-hard problem with applications in such diverse areas as VLSI layout design, program design for paged computer memory, group technology analysis, qualitative data analysis, image analysis, and cluster analysis. Many other applications, such at the formation of alliances among countries as well as strategic alliances among companies, can also be modeled and analyzed as clique partitioning problems.

2.1. Standard 0/1 linear model

The standard formulation of CP (see for instance, Grotschel and Wakabayashi, 1989; Chopra and Rao, 1993; Oosten et al., 2001) is given by:

$$\text{CP(Edge)} : \max \quad x_0 = \sum_{(i,j)\in E} w_{ij} x_{ij} \tag{1}$$

$$\text{st} \quad x_{ij} + x_{ir} - x_{jr} \leq 1 \ \ \forall \ \textit{distinct} \ \ i, j, r \ \in V$$
$$x_{ij} \in \{0, 1\} \tag{2}$$

where the w_{ij} are unrestricted edge weights and $\boldsymbol{x}_{ij}$ is defined to be 1 if edge (i, j) is in the partition, and equal to 0 otherwise. This *edge-based* formulation contains $\boldsymbol{n(n-1)/2}$ variables and $3\boldsymbol{C}_3^n$ constraints and the model explodes in size even for modest sized graphs,. Despite these model limitations, the dominant methods presented in the literature for solving CP[Edge] are exact approaches based on LP methods as illustrated by the cutting plane approaches of Grotschel and Wakabayashi (1989) and Oosten et al. (2001), and the column generation approach of Mehrotra and Trick (1998). These approaches have proven to be successful on small to moderate size problems. For larger instances, however, their application is severely limited due the challenge presented by the large size of CP[Edge].

For such cases, metaheuristic methods, coupled with a new formulation, prove to be very effective as we demonstrate in the following sections.

2.2. *New formulation*

The computational challenge posed by CP[Edge] for large problem instances motivates the development of a new formulation that can be readily solved by basic metaheuristic methodologies. We first present the new model and then describe our solution approach.

Without loss of generality we assume G is a complete graph, where artificial edges with negative (penalty) edge weights are introduced as needed to assure an edge exists between each pair of nodes. In addition, define

$\boldsymbol{k\_max}$ = maximum number of cliques allowed (an educated guess)

and

$\boldsymbol{x}_{ik} = 1$ if node i is assigned to clique k; 0 otherwise

Then our model is:

$$\text{CP[Node]:} \quad \max \quad x_0 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} \sum_{k=1}^{k\_\max} x_{ik} x_{jk} \tag{3}$$

$$\text{st} \quad \sum_{k=1}^{\boldsymbol{k\_max}} x_{ik} = 1 \quad i = 1, n \tag{4}$$

The quadratic terms in the objective function imply that the weight $\boldsymbol{w}_{ij}$ becomes part of the partition weight only when nodes i and j are assigned to the same clique. The constraints of (4) require that each node is assigned to one of the cliques.

The value chosen for k_max for a given problem instance can be estimated from domain knowledge. In determining k_max, we bias the estimation slightly toward the *high* side while trying not to make it unnecessarily large as doing so inflates the number of variables in the model. In the event that k_max is inadvertently set too low, the bound that it represents will be achieved, denoting that k_max needs to be increased and the model solved again. While it is plausible that such an iterative procedure may be required in a given case, our experience suggests that this happens infrequently. Over a wide variety of test problems, setting an appropriate value for k_max has not been a problem.

Several remarks about this model are in order: First of all, this node-oriented model contains many fewer variables than CP[Edge] since $\boldsymbol{n}(\boldsymbol{k\_max})$ is typically much less than $n(n-1)/2$. Furthermore, the number of constraints (n) is much smaller than the corresponding number ($3\boldsymbol{C}_3^n$) for the edge-oriented model of CP[Edge]. While CP[Edge] is a linear model and CP[Node] is quadratic, the size difference enables this quadratic alternative to be used for large instances of clique partitioning problems where the computational burden of CP[Edge] precludes its practical use. As we demonstrate later in this paper, CP[Node]

can be effectively solved, even for large instances, by modern metaheuristic methods such as tabu search.

2.2.1. Solving CP[Node]. Our approach to solving CP[Node] is to first re-cast the model into the form of either an unconstrained binary quadratic program (UBQP) or a very slight variation of UBQP consisting of a cardinality constrained binary quadratic program (CBQP). Our motivation is to leverage the advances we have reported elsewhere in the recent literature for solving UBQP, a method embodying basic tabu search features that is capable of solving our revised CP[Node] model as represented by either UBQP or CBQP. We start here with the development for UBQP and comment later about the slight variation CBQP.

Our node formulation, CP[Node], can be represented in matrix notation as:

$$\begin{aligned} &\max x_o = xQx \\ \text{suject to} \quad &Ax = b, x \text{ binary} \end{aligned} \tag{5}$$

Such models can be converted into equivalent unconstrained models (UBQP) by adding a quadratic infeasibility penalty function to the objective function in place of explicitly imposing the constraints A$x = b$.

Specifically, for a positive scalar P, we have

$$\begin{aligned} x_0 &= xQx - P(Ax - b)^t(Ax - b) \\ &= xQx - xDx - c \\ &= x\hat{Q}x - c \end{aligned} \tag{6}$$

where the matrix D and the additive constant c result directly from the matrix multiplication indicated. Dropping the additive constant, the equivalent unconstrained version of our constrained problem becomes

$$\text{UBQP} : \max x\hat{Q}x, x \text{ binary} \tag{7}$$

From a computational standpoint, a suitable choice of the penalty scalar P can always be chosen so that the optimal solution to UBQP is the optimal solution to the original constrained problem. This approach has proven to be successful for a wide variety of problem classes as reported in Kochenberger et al. (2004) and Kochenberger and Glover (2005).

2.2.2. Solving UBQP. The reformulated version of CP[Node] can be efficiently solved by the tabu search method described in Glover et al. (1999). An overview of this method is as follows.

Our *TS* method for UBQP is based on the use of strategic oscillation, which constitutes one of the primary strategies of tabu search. The variant of strategic oscillation we employ may be sketched in overview as follows.

The method alternates between constructive phases that progressively set variables to 1 (whose steps we call "add moves") and destructive phases that progressively set variables

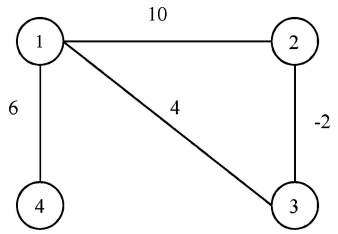
to 0 (whose steps we call "drops moves"). To control the underlying search process, we use a memory structure that is updated at *critical events*, identified by conditions that generate a subclass of locally optimal solutions. Solutions corresponding to critical events are called *critical solutions*.

A parameter *span* is used to indicate the amplitude of oscillation about a critical event. We begin with *span* equal to 1 and gradually increase it to some limiting value. For each value of *span*, a series of alternating constructive and destructive phases is executed before progressing to the next value. At the limiting point, *span* is gradually decreased, allowing again for a series of alternating constructive and destructive phases. When *span* reaches a value of 1, a *complete span cycle* has been completed and the next cycle is launched.

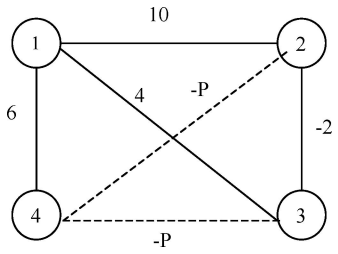
Information stored at critical events is used to influence the search process by penalizing potentially attractive add moves (during a constructive phase) and inducing drop moves (during a destructive phase) associated with assignments of values to variables in recent critical solutions. Cumulative critical event information is used to introduce a subtle long term bias into the search process by means of additional penalties and inducements similar to those discussed above.

We illustrate our approach to clique partitioning via CP[Node] and UBQP by the following example:

Example 1. Consider the graph



Adding artificial edges (2,4) and (3,4) with edge weights equal to –P, we get the complete graph



Arbitrarily taking k_max to be 3 (a choice whose appropriateness can be established from the solution to the model), CP[Node] becomes:

$$\begin{aligned}
\max x_o = {} & 10x_{11}x_{21} + 4x_{11}x_{31} + 6x_{11}x_{41} - 2x_{21}x_{31} - Px_{21}x_{41} - Px_{31}x_{41} \\
& +10x_{12}x_{22} + 4x_{12}x_{32} + 6x_{12}x_{42} - 2x_{22}x_{32} - Px_{32}x_{42} - Px_{32}x_{42} \\
& +10x_{13}x_{23} + 4x_{13}x_{33} + 6x_{13}x_{43} - 2x_{23}x_{33} - Px_{23}x_{43} - Px_{33}x_{43} \\
\text{st} \quad & x_{11} + x_{12} + x_{13} = 1 \\
& x_{21} + x_{22} + x_{23} = 1 \\
& x_{31} + x_{32} + x_{33} = 1 \\
& x_{41} + x_{42} + x_{43} = 1
\end{aligned} \tag{8}$$

Choosing $P = 20$ and applying the quadratic infeasibility transformation of (6) yields the equivalent unconstrained problem:

$$\text{UBQP: } \max \hat{x}_0 = x\hat{Q}x \tag{9}$$

with an additive constant of -80 and the $\hat{Q}$ matrix is given by:

$$\begin{bmatrix}
20 & -20 & -20 & 5 & 0 & 0 & 2 & 0 & 0 & 3 & 0 & 0 \\
-20 & 20 & -20 & 0 & 5 & 0 & 0 & 2 & 0 & 0 & 3 & 0 \\
-20 & -20 & 20 & 0 & 0 & 5 & 0 & 0 & 2 & 0 & 0 & 3 \\
5 & 0 & 0 & 20 & -20 & -20 & -1 & 0 & 0 & -10 & 0 & 0 \\
0 & 5 & 0 & -20 & 20 & -20 & 0 & -1 & 0 & 0 & -10 & 0 \\
0 & 0 & 5 & -20 & -20 & 20 & 0 & 0 & -1 & 0 & 0 & -10 \\
2 & 0 & 0 & -1 & 0 & 0 & 20 & -20 & -20 & -10 & 0 & 0 \\
0 & 2 & 0 & 0 & -1 & 0 & -20 & 20 & -20 & 0 & -10 & 0 \\
0 & 0 & 2 & 0 & 0 & -1 & -20 & -20 & 20 & 0 & 0 & -10 \\
3 & 0 & 0 & -10 & 0 & 0 & -10 & 0 & 0 & 20 & -20 & -20 \\
0 & 3 & 0 & 0 & -10 & 0 & 0 & -10 & 0 & -20 & 20 & -20 \\
0 & 0 & 3 & 0 & 0 & -10 & 0 & 0 & -10 & -20 & -20 & 20
\end{bmatrix} \tag{10}$$

Solving UBQP by our tabu search code gives $\hat{x}_0 = 92$ for the nonzero assignments $x_{11} = x_{21} = x_{31} = x_{42} = 1$. This optimal solution to CP[Node] consists of two cliques with an objective function value of $x_0 = 92 - 80 = 12$ with nodes 1, 2 and 3 assigned to one clique and node 4 assigned to a second clique. Note that while we allowed for the possibility of three cliques, only two were used in our solution. This implies that our initial choice of k_max = 3 was more than sufficient. In general, whenever the initial value selected for k_max yields a solution that contains fewer than k_max cliques, the chosen value is vindicated. Otherwise, we only need to increase k_max and repeat the solution process.

Earlier in this section we commented that the cardinality constrained binary quadratic program, CBQP, could be employed as an alternative reformulation of CP[Node]. This slight variation of UBQP takes the form:

$$\begin{aligned} &\text{CBQP:} \quad \max \quad x\bar{Q}x \\ &\text{st} \quad \sum x_{ij} = n \end{aligned} \tag{11}$$

The differences between the models put forth in (6) and (10) are slight. CBQP does not have the additive constant of UBQP but has a single cardinality constraint requiring that exactly n of the variables are equal to 1. The "Q" matrices in the two formulations differ only in their main diagonals where the diagonal elements in UBQP contain penalty terms and those of CBQP do not. These slight variations, equally solvable by our tabu search method, are equivalent reformulations of CP[Node]. Our computational testing suggests that they are both effective with neither offering a computational advantage over the other.

In Section 2.3 below we present computational experience comparing CP[Edge] and CP[Node] for some test problems of moderate to large size. Our results for CP[Edge] were produced by CPLEX and those given for CP[Node] were produced by our tabu search method addressing CP[Node] via CBQP.

2.3. Computational comparisons of models

To provide a basis for comparing CP[Edge] and CP[Node], test problems of size $n = 25$, 50, 100, and 200 (three instances in each case) were generated and solved. For each of the 12 problems, edge weights were randomly generated with an absolute magnitude between 1 and 50. Roughly 30% of the edge weights were then made negative.

Table 1 shows the results obtained by applying CPLEX to CP[Edge]. The corresponding results for CP[Node] are given in Table 2.

All runs were carried out using a Sun Enterprise 3500 server. The results of Table 1 were obtained from CPLEX 6.5 and those of Table 2 were obtained from our tabu search heuristic with an arbitrary limit of 100 SPAN cycles for each problem. The first three columns of both tables are identical denoting the problem ID along with the number of nodes and edges in the graph. Columns 4 and 5 of Table 1 denote the number of variables and constraints in the CP[Edge] formulation and the last two columns give the objective function values and corresponding solution times.

The fourth column of Table 2 gives the maximum number of cliques allowed and the fifth column gives the number of variables in the CBQP formulation. The sixth and seventh columns give the objective function value and number of cliques actually formed in the solution, respectively. Finally, the last column gives the time taken for the heuristic to perform the 100 SPAN cycles.

Note that CPLEX was able to terminate naturally on the first three problems only. For these same three problems, our tabu search method gave the same (i.e., optimal) solution in a fraction of the time required by CPLEX. For the 50 node problems, CPLEX was allowed to run for an arbitrary limit of 40 CPU hours. The results given in Table 1 for these problems are the best solutions found during the 40 hour run. As shown in Table 2, our tabu

Table 1. CPLEX results for CP[Edge].

ID	# Nodes	# Edges	# Vars	# Const	Soln	Time (seconds)
25.1	25	300	300	6900	2143	776
25.2	25	300	300	6900	2361	607
25.3	25	300	300	6900	1547	1191
50.1	50	1225	1225	58800	2926**	T_lim
50.2	50	1225	1225	58800	3103**	T_lim
50.3	50	1225	1225	58800	3678**	T_lim
100.1	100	4950			(NA)	
100.2	100	4950			(NA)	
100.3	100	4950			(NA)	
200.1	200	19,900			(NA)	
200.2	200	19,900			(NA)	
200.3	200	19,900			(NA)	

Table 2. CP[Node] results via tabu search.

ID	# Nodes	# Edges	k_max	# Vars	Soln	# Cliques	Time (seconds)
25.1	25	300	5	125	2143	3	1.3
25.2	25	300	5	125	2361	3	1.3
25.3	25	300	5	125	1547	3	1.2
50.1	50	1225	5	250	4884	3	2.9
50.2	50	1225	5	250	5127	3	2.9
50.3	50	1225	5	250	5282	3	2.9
100.1	100	4950	5	500	19929	3	7.4
100.2	100	4950	6	600	16588	5	9.6
100.3	100	4950	6	600	22541	3	8.1
200.1	200	19,900	6	1200	56125	5	33
200.2	200	19,900	6	1200	55030	4	33
200.3	200	19,900	6	1200	59441	5	38

search required less than 3 seconds to perform 100 SPAN cycles and returned solutions significantly better than those found by CPLEX for these $n = 50$ problems.

The $n = 100$ and $n = 200$ instances were simply too large for CPLEX as the LP relaxations are either too large to be solved (at all) or too large to be solved in a reasonable amount of time (for the $n = 100$ case). As a result, applying CPLEX to the edge-based model was unable to produce solutions for these problems. In contrast to this, our tabu search method, as shown in Table 2, readily produced solutions for even the largest of these problems in less than 40 seconds of computational time.

3. Clique partitioning and clustering

To use the clique partitioning model as a tool for clustering, we undertake to form cliques on a similarity or proximity graph. This approach to clustering was championed by Grotschel and Wakabayashi (1989) and later by Dorndorf and Pesch (1994) and Mehrotra and Trick (1998). In each case, these authors advocated the use of the edge-based model CP[Edge] and good results were reported on some small examples. The applicability of this model to larger clustering problems, such as those encountered in the mining of Microarray data, is greatly restricted due to the computational difficulties mentioned in Section 2. Our node-based model, however, overcomes this size limitation and thus enables the clique partitioning approach to be applied even in these more challenging settings. To apply CP[Node] to the clustering of Microarray Gene data, we proceed as follows:

We represent the problem by a proximity graph with a node for each gene. Edge weights are computed by first computing a standardized "distance" matrix denoting the similarity of gene expression results for each pair of genes. From this matrix a *threshold* distance is computed and this threshold amount is used to produce edge weights via

$$w_{ij} = 100^{*}(d(i, j) - \text{threshold}) \tag{12}$$

where $d(i, j)$ is the standardized distance between genes i and j and *threshold* is the threshold value. Given this graph, we solve CP[Node] for the objective of minimizing the total weight across all cliques in the partition. While it is possible to define the threshold value in a variety of ways, we have found that taking the threshold to be the average of the $d(i, j)$ values yields very good results.

Example 2. This approach is illustrated by a small example taken from Shannon et al. (2003) consisting of the following gene expression data for four genes and two chips:

Gene	Chip 1	Chip 2
A	−2.0	1.0
B	−1.5	−0.5
C	1.0	0.25
D	2.5	2.5

For this raw data, a standardized Euclidean distance matrix is given by

	A	B	C	D
A	0.000	0.512	0.712	1.118
B	0.512	0.000	0.609	1.338
C	0.712	0.609	0.000	0.821
D	1.118	1.338	0.821	0.000

We compute a threshold (average distance) value of 0.852 and obtain the following edge weights:

Node	Node	Weight
A	B	−34
A	C	−14
A	D	27
B	C	−24
B	D	49
C	D	−3

Taking K_max to be 2, CP[Node] becomes:

$$\begin{aligned}
\text{Min} \; & -34(x_{11}x_{21} + x_{12}x_{22}) - 14(x_{11}x_{31} + x_{12}x_{32}) + 27(x_{11}x_{14} + x_{12}x_{42}) \\
& -24(x_{12}x_{31} + x_{22}x_{32}) + 49(x_{21}x_{41} + x_{22}x_{42}) - 3(x_{31}x_{41} + x_{32}x_{42}) \\
\text{st} \quad & x_{11} + x_{12} = 1 \\
& x_{21} + x_{22} = 1 \\
& x_{31} + x_{32} = 1 \\
& x_{41} + x_{42} = 1
\end{aligned} \tag{13}$$

This instance of CP[Node] is solved by re-casting it in the form of

$$\begin{aligned}
&\text{CBQP:} \quad \min x'Qx \\
&\text{st} \\
&\sum x_{ij} = 4
\end{aligned} \tag{14}$$

Where the Q matrix is given by (for a scalar infeasibility penalty of 99):

$$\begin{bmatrix}
0 & 99 & -34 & 0 & -14 & 0 & 27 & 0 \\
99 & 0 & 0 & -34 & 0 & -14 & 0 & 27 \\
-34 & 0 & 0 & 99 & -24 & 0 & 49 & 0 \\
0 & -34 & 99 & 0 & 0 & -24 & 0 & 49 \\
-14 & 0 & -24 & 0 & 0 & 99 & -3 & 0 \\
0 & -14 & 0 & -24 & 99 & 0 & 0 & -3 \\
27 & 0 & 49 & 0 & -3 & 0 & 0 & 99 \\
0 & 27 & 0 & 49 & 0 & -3 & 99 & 0
\end{bmatrix}$$

Solving CBQP yields two clusters (ABC, D).

4. Computational experience

To test our model on real data, we obtained six sets of Microarray data from St. Jude Children's Research Hospital in Memphis, Tennessee. These data sets contain gene expression data obtained from bone marrow samples from acute lymphoblastic leukemia patients. In the results given below, we compare CP[Node] with the commonly used K-means method of SAS 9.1. In each case, CP[Node] is re-cast in the form of CBQP and solved by our tabu search method.

One of the draw-backs of K-means is that the number of clusters (K) needs to be specified in advance and thus a series of runs, each with a different value for K, may be needed in order to identify ideal results. Our model, in contrast, requires only an upper bound on the

Table 3. Computational experience on different St. Jude "ALL" data sets.

			CP method				*k*-means (SAS 9.1)		
Dataset	# of genes	# of chips	# of var	# of clusters	# of outliers	CPU time (second)	# of clusters	# of outliers	CPU time (second)
BCR	304	15	2432	7	14	30	7	11	14
E2A	304	27	2432	5	7	30	5	5	11
MLL	304	20	3648	11	12	49	11	12	15
Hyperdip50	304	64	2432	8	14	30	8	12	18
T	304	43	3648	11	14	49	11	10	21
TEL	304	79	2432	5	8	30	5	4	12

Table 4. Clustering result on St Jude ALL(BCR) data set.

Cluster #1	*k*-means	CP	Genes in common
# genes in cluster	293	290	288
Cluster #2	(1, 2, 5, 6)	(1, 2, 5, 7, 8, 10)	
# genes in cluster	4	6	3
Cluster #3	(3, 4, 14)	(3, 4, 6, 14)	
# genes in cluster	3	4	3
Cluster #4	(9)	(9)	
# genes in cluster	1	1	1
Cluster #5	(11)	(11)	
# genes in cluster	1	1	1
Cluster #6	(12)	(12)	
# genes in cluster	1	1	1
Cluster #7	(13)	(13)	
# genes in cluster	1	1	1

number of clusters and finds the optimal number of clusters less than or equal to this bound. Provided that the bound is sufficiently large, only one run of our model is needed as opposed to several for K-means. Moreover, we immediately know that the bound has been properly selected when the number of cliques identified in our solution is less than this bound.

Table 5. Clustering result on St Jude ALL (T) data set.

Cluster #1	*k*-means	CP	# Genes in common
# genes in cluster	160	152	151
Cluster #2			
# genes in cluster	133	138	131
Cluster #3	(1, 2)	(1, 2, 10)	
# genes in cluster	2	3	2
Cluster #4	(3)	(3, 7, 8)	
# genes in cluster	1	3	1
Cluster #5	(9)	(9,14)	
# genes in cluster	1	2	1
Cluster #6	(4)	(4)	
# genes in cluster	1	1	1
Cluster #7	(5)	(5)	
# genes in cluster	1	1	1
Cluster #8	(11)	(11)	
# genes in cluster	1	1	1
Cluster #9	(12)	(12)	
# genes in cluster	1	1	1
Cluster #10	(13)	(13)	
# genes in cluster	1	1	1
Cluster #11	(6)	(6)	
# genes in cluster	1	1	1

Table 6. Clustering Result on St Jude ALL (E2A) Data Set.

Cluster #1	*k*-means	CP	# Genes in common
# genes in cluster	258	260	255
Cluster #2			
# genes in cluster	41	38	36
Cluster #3	(4,9,12)	(2,4,7,9,12)	
# genes in cluster	3	5	3
Cluster #4	(11)	(11)	
# genes in cluster	1	1	1
Cluster #5	(13)	(13)	
# genes in cluster	1	1	1

In order to provide a comparison that is as fair as possible and to avoid having to make multiple runs of K-means, we first solved each of the six data sets with our method to determine the optimal number of clusters needed. This number of clusters, then was taken as K for the subsequent runs on the data sets via SAS's K-means procedure. In this manner, we are able to compare clustering results for each data set based on the same number of clusters. The overall results from these runs are summarized in Table 3. All computations were carried out on a Sun 420 R workstation equipped with four UltraSparcII CPUs. Since our method employs a metaheuristic solution methodology, and thus does more work than the K-means heuristic, our solution times, as expected, are larger than those of K-means. Nevertheless, in all cases, the computation times are modest.

Table 7. Clustering result on St Jude ALL (TEL) data set.

Cluster #1	*k*-means	CP	# Genes in common
# genes in cluster	266	260	260
Cluster #2			
# genes in cluster	34	36	31
Cluster #3	(2, 4)	(1, 2, 4, 9)	
# genes in cluster	2	4	2
Cluster #4	(12)	(11, 12, 6)	
# genes in cluster	1	3	1
Cluster #5	(13)	(13)	
# genes in cluster	1	1	1

Table 8. Clustering result on St Jude ALL(Hyperdip50) data set.

Cluster #1	*k*-means	CP	# Genes in common
# genes in cluster	262	152	150
Cluster #2			
# genes in cluster	30	138	28
Cluster #3	(8, 10, 11, 12)	(1, 2, 3, 8, 10, 11, 12)	
# genes in cluster	4	7	4
Cluster #4	(7, 14, 63)	(7,14)	
# genes in cluster	3	2	2
Cluster #5	(4, 9)	(4, 9)	
# genes in cluster	2	2	2
Cluster #6	(6)	(6)	
# genes in cluster	1	1	1
Cluster #7	(5)	(5)	
# genes in cluster	1	1	1
Cluster #8	(13)	(13)	
# genes in cluster	1	1	1

In Table 3 we report *outliers* as genes in clusters containing 6 or fewer members. This definition is somewhat arbitrary. Nonetheless the point is made in the table that the composition of the clusters produced by the two models with respect to outliers differed on five of the six data sets for this definition. As noted in subsequent tables, clusters of smaller size, containing one or two genes, are much more uniform.

Tables 4–9 give detailed cluster results for each of the six data sets examined in this paper. In each case we see a pattern of one or two large clusters augmented by several small clusters. For each data set (table) we report the number of genes in each cluster, as produced by each method, along with the number of genes in common for each cluster. For small clusters, we report the actual genes that were put in the various clusters. For large clusters, we simply report the aggregate number of genes.

For instance, Table 4 reports the results obtained for the BCR data where the 304 genes were assigned to one of 7 clusters. For both K-means and CP, the results show one large cluster and 6 small clusters. The K-means procedure produced Cluster # 1 consisting of 293 genes and our CP model produced Cluster # 1 with 290 genes, 288 of which were common to both clusters. For K-means, cluster # 2 consists of genes 1, 2, 5, and 6 (i.e., 4 genes) and

Table 9. Clustering result on St Jude ALL(MLL) data set.

Cluster #1	*k*-means	CP	# Genes in common
# genes in cluster	266	262	261
Cluster #2			
# genes in cluster	14	19	12
Cluster #3	(2,3,6,20,103,286,294)	(6,7,20,103,286,294)	
# genes in cluster	7	6	5
Cluster #4	(1,48,117,132,151)	(1,48,117,132,151)	
# genes in cluster	5	5	5
Cluster #5	(8,141,236)	(8,141,236)	
# genes in cluster	3	3	3
Cluster #6	(9,148,235)	(9,148,235)	
# genes in cluster	3	3	3
Cluster #7	(5,84)	(5,84)	
# genes in cluster	2	2	2
Cluster #8	(4)	(4)	
# genes in cluster	1	1	1
Cluster #9	(13)	(13)	
# genes in cluster	1	1	1
Cluster #10	(11)	(11)	
# genes in cluster	1	1	1
Cluster #11	(12)	(12)	
# genes in cluster	1	1	1

the corresponding cluster for CP consists of genes 1, 2, 5, 7, 8, and 9 for a total of 6 genes, three of which are in common with the corresponding K-means cluster.

As the tables indicate, across all six data sets, the clusters produced are fairly similar in general composition, but some substantial differences are seen in roughly 50% of the clusters as we compare corresponding clusters produced by the two methods. These differences are most pronounced for the larger clusters that are formed with the most extreme cases coming from the Hyperdip50 data set (Table 8) where clusters 1 and 2, as produced by K-means and CP, exhibit fundamental differences in both size and composition.

The results displayed in Tables 3–9 suggest that our CP model produced clusters that are generally similar in size to those produced by the K-means procedure for each data set but exhibit considerable differences in the actual gene composition of many of the clusters formed. The implication of these differences is currently under investigation.

5. Summary

In this paper we have introduced a new model for clique partitioning (CP) that extends the usefulness of CP as a methodology for clustering data. This model is particularly appropriate in application settings, as represented by Microarray data, where the data sets are too large to be accommodated by the standard CP formulation. Computational experience shows the superiority of the new model (CP[Node]) compared to the standard edge-based model (CP[Edge]) on medium to large instances of clique partitioning. Computational experience is also presented showing the performance of our new model for the problem of clustering Microarray data, making reference to outcomes obtained from SAS's K-means procedure to provide a benchmark of comparison.

A noted feature of our approach is that the optimal number of clusters, given an appropriate upper bound, is computed by the model. This is a significant advantage over alterative methods where the number of clusters must be specified in advance.

The results presented show that our new model is computationally attractive for large clique partitioning problems and exhibits considerable potential as a tool for clustering. The metaheuristic method used to solve our new model, by its very nature, is designed to produce a more robust search process than that coming from a simpler heuristic such as K-means. As a result, the differences reported in the clustering results for the Microarray data are not unexpected. The extent to which these differences may have significant implications from a functional genomic/molecular biology perspective is currently under investigation.

Acknowledgments

The authors would like to thank the two anonymous referees for their useful comments and suggestions.

References

S. Chopra and M.R. Rao, "The partition problem," *Mathematical Programming*, vol. 59, pp. 87–115, 1993.

U. Dorndorf and E. Pesch, "Fast clustering algorithms," *ORSA J. Comput.*, vol .6, pp. 141–153, 1994.

F. Glover, G. Kochenberger, B. Alidaee, and M. Amini, "Tabu search with critical event memory: An enhanced application for binary quadratic programs," in *Meta-Heuristics, Advances and Trends in Local Search Paradigms for Optimization*, S.M.S. Voss, I. Osman, and C. Roucairol (Eds.), Kluwer Publisher, 1999, pp. 93–109.

M. Grotschel and Y. Wakabayashi, "A cutting plane algorithm for a clustering problem," *Mathematical Programming*, vol. 45, pp. 59–96, 1989.

M. Grotschel and Y. Wakabayashi, "Facets of the clique partitioning polytope," *Mathematical Programming*, vol. 47, pp. 367–387, 1990.

D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: A survey," *IEEE Transactions on Knowledge & Data Engineering*, vol. 16, pp. 1370–1386, 2004.

G. Kochenberger, F. Glover, B. Alidaee, and C. Rego, "A unified modeling and solution framework for combinatorial optimization problems," *OR Spectrum*, vol. 26, pp. 237–250, 2004.

G. Kochenberger and Fred Glover, "A unified framework for modeling and solving combinatorial optimization problems: A tutorial," in *Multiscale Optimization Methods and Applications*, W. Hager and P. Pardalos (Eds.), (to be published by Springer in 2005).

A. Mehrotra and M. Trick, "Cliques and clustering: A combinatorial approach," *Operations Research Letters*, vol. 22, pp. 1–12, 1998.

M. Oosten, J. Rutten, and F. Spieksma, "The clique partitioning problem: Facets and patching facets," *Networks*, vol. 38, pp. 209–226, 2001.

W. Shannon, R. Culverhouse, and J. Duncan, "Analyzing microarray data using cluster analysis," *Pharmacogenomics*, vol. 4, pp. 41–52, 2003.