Mathematical Programming 62 (1993) 133–151
North-Holland

Min-cut clustering

Ellis L. Johnson, Anuj Mehrotra and George L. Nemhauser

School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA

Received 12 March 1992
Revised manuscript received 17 August 1992

This paper is dedicated to Phil Wolfe on the occasion of his 65th birthday.

We describe a decomposition framework and a column generation scheme for solving a min-cut clustering problem. The subproblem to generate additional columns is itself an NP-hard mixed integer programming problem. We discuss strong valid inequalities for the subproblem and describe some efficient solution strategies. Computational results on compiler construction problems are reported.

Key words: Clustering, decomposition, column generation, subproblem optimization, valid inequality, compiler design.

1. Introduction and formulation

Given an undirected graph $G(V, E)$ with nonnegative weights w_v, $v \in V$, edge costs c_e, $e \in E$, and an integer K, the clustering problem is to find a partition $\Gamma = \{W_1, W_2, \ldots, W_K\}$ of V that solves

$$\text{Max(Min)} \sum_{i=1}^{K} \sum_{e \in E(W_i)} c_e,$$

$$w_{\min} \leqslant \sum_{u \in W_k} w_u \leqslant w_{\max}, \quad k = 1, 2, \ldots, K,$$

where $E(W_i) = \{(i, j) \in E: i, j \in W_i\}$. We will refer to the W_i, $i = 1, \ldots, K$, as *clusters* corresponding to the partition Γ of V. In other words, the clustering problem is to partition the nodes of the graph into K clusters such that the sum of the node weights of each cluster is bounded from below by $w_{\min}$ and bounded from above by $w_{\max}$ while maximizing (minimizing) the sum of the costs on the edges inside clusters.

Our study differs primarily in three ways from most of the papers in the clustering literature. We do not assume that the underlying graph is complete; we include a constraint on the size or weight of the clusters and we solve these clustering problems using a decomposition and column generation scheme.

Correspondence to: George L. Nemhauser, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332–0205, USA.

This research was supported by NSF grants DMS-8719128 and DDM-9115768, and by an IBM grant to the Computational Optimization Center, Georgia Institute of Technology.

Given a graph $G(V, E)$ and a partition $\Gamma = \{W_1, W_2, \ldots, W_K\}$ of V, the set of edges $\{(i, j): i \in W_p, j \in W_q, p \neq q\}$ is called a *multicut*. Maximizing (minimizing) the sum of the costs on edges within the clusters is equivalent to minimizing (maximizing) the sum of costs on the edges that go between clusters, that is the edges in a multicut. When all the edge costs are nonnegative, we will refer to these problems as *min-cut clustering* (MCC) and *max-cut clustering* problems respectively. When the edge costs are not restricted to be nonnegative, we will call them *mixed-cut clustering* problems. MCC is also referred to as the *graph partitioning* problem.

All the clustering problems described above are known to be NP-hard. For a survey of the complexity of related problems, see [7] and [8]. The clustering problems defined above generalize the definition of clustering problems encountered in the literature and encompass several well-known combinatorial problems such as the min-cut problem, the max-cut problem, the clique partitioning problem, the equipartition problem, and the unconstrained quadratic 0–1 programming problem. Most of these special cases of the generalized clustering problem are themselves known to be NP-hard.

Ignoring the weight restrictions on the clusters and fixing $K=2$ yields the well known *min(max)-cut* problems. Note that while *min-cut* is solvable in polynomial time, *max-cut* is NP-hard. If $K=2$, and there is a restriction that both the clusters must have the same (or within 1 if $|V|$ is odd) number of nodes, the problem is the NP-hard *equipartition* problem studied by Conforti et al. [5] and [6].

A set of edges A in a graph $G(V, E)$ is called a *clique partitioning* of G if there is a partition $\Gamma = \{W_1, W_2, \ldots, W_k\}$ of V such that $A = E(W_i) \cup E(W_2) \cup \cdots \cup E(W_k)$ and such that the subgraph $G[W_i]$ induced by W_i is complete for $i=1, 2, \ldots, k$. Given a graph $G(V, E)$ with costs $c_e \in \mathbb{R}$ for $e \in E$, the clique partitioning problem (CPP) is to find a clique partitioning $A \subseteq E$ of minimum cost. Grötschel and Wakabayashi [11] have studied the CPP on complete graphs. The CPP on a complete graph is an example of a *mixed-cut* clustering problem with no restrictions on the number or size/weight of clusters.

Chopra and Rao have studied MCC without the weight restrictions, see [2–4]. Chopra [2] shows that if the graph is series-parallel, the associated polyhedron can be completely described.

In our solution approach of column generation and subproblem optimization, the subproblem of generating one feasible cluster is a generalization of the *Boolean quadratic program* studied by Padberg [16]. The Boolean quadratic program is an integer programming formulation of the problem of maximizing (minimizing) an unconstrained quadratic 0–1 function.

We now restrict our discussion to MCC. One application of MCC is in compiler construction. A compiler consists of several modules, where each module is a set of procedures or subroutines with its associated storage/memory requirement. The modules are combined to form clusters which are restricted in their total storage. A typical storage bound on each cluster is either $450K$ or $512K$. The communication costs between modules within the same cluster are negligible and can therefore be

ignored. The communication costs between modules of different clusters are substantial because the clusters may be required to be swapped in memory. The objective of compiler construction is to assign the modules to clusters so that the storage bound restriction is satisfied and the total communication cost between modules in different clusters is minimized. In the framework of MCC, the modules are represented by nodes with the storage requirement of each module represented by the weight on the corresponding node. The communication cost between modules is represented by the cost on the edge between the corresponding nodes. Here the number of clusters to be formed is not fixed.

Next, we present a straightforward integer programming formulation of MCC. Let $x_i^k = 1$ if $i \in$ *cluster* k, and 0 otherwise. Let $z_e^k = 1$ if edge e belongs to cluster k, and 0 otherwise. Assume for simplicity of exposition, that the number of clusters to be formed is K and that the lower bound on the weight of every cluster is 0 while the upper bound on the weight of every cluster is the same and is b. Then the problem may be formulated as follows.

$$\text{Max} \sum_{k=1}^{K} \sum_{e \in E} c_e z_e^k, \tag{1}$$

$$z_e^k \leqslant x_i^k, \quad z_e^k \leqslant x_j^k, \qquad e = (i,j) \in E, \tag{2}$$

$$z_e^k \geqslant x_i^k + x_j^k - 1, \quad e = (i,j) \in E, \tag{3}$$

$$\sum_{k=1}^{K} x_i^k = 1, \quad i = 1, \ldots, |V|, \tag{4}$$

$$\sum_{i \in V} x_i^k \geqslant 1, \quad k = 1, \ldots, K, \tag{5}$$

$$\sum_{i \in V} w_i x_i^k \leqslant b, \quad k = 1, \ldots, K, \tag{6}$$

$$\text{all variables binary.} \tag{7}$$

Inequalities (2) and (3) ensure that an edge is inside a cluster if and only if both of its end nodes are in that cluster. Inequality (4) ensures each node to be in some cluster, (5) ensures that no cluster is empty, that is there are K clusters and not fewer, and (6) provides the weight restriction on each cluster and is referred to as the knapsack constraint. Note that the edge variables need not be restricted to be integers because they will automatically be so in any optimal solution where the node variables are integers. Furthermore, since the edge costs are nonnegative, (3) can be dropped because an optimal solution will automatically satisfy it. If the number of clusters is not fixed, K is set equal to $|V|$ and (5) is omitted, we refer to (1), (2), and (4)–(7) as the binary min-cut clustering formulation. Its linear programming relaxation in which (7) is replaced by nonnegativity on all variables is denoted by LPMCC.

In the next section, we present a stronger formulation of MCC that uses a decomposition based solution strategy which creates a master problem and a subproblem to generate columns for the master problem. In Section 3, we discuss the issues

related to subproblem optimization. In Section 4, we discuss implementation issues and report computational results.

2. Decomposition and column generation

Setting all the variables to $1/K$ provides the optimal solution to LPMCC with the best possible objective value of $\sum_{e\in E} c_e$. Thus, LPMCC provides a useless bound on the integer programming optimal value. An attempt to solve MCC directly (using the formulation above) via a branch-and-bound scheme may be fruitless for any practical problem size. This is largely due to the weakness of the LP relaxation and the inherent symmetry in the formulation. Using symmetry, any partition of the nodes into clusters can be represented by many equivalent solutions by exchanging the cluster numbers of any two clusters. Fixing a node variable $x_i^1 = 0$ still leaves $x_i^2, \ldots, x_i^K$ nonzero. That is, fixing of variables in a branch-and-bound solution procedure is not efficient because several alternate optimal solutions may still be feasible.

We next describe a different formulation for MCC that uses a decomposition based solution strategy and column generation. This Dantzig–Wolfe type of decomposition strategy is discussed here specifically in the context of min-cut clustering problems, but can be applied to more general MIP problems as suggested in [12–14].

2.1. Stronger formulation

A cluster is feasible if the weights on the nodes in the cluster satisfy the knapsack constraint (6). Let y^l, $l = 1, 2, \ldots, L$ be the incidence vectors of all possible feasible clusters, where $y_i^l = 1$ if node i is in cluster l, and 0 otherwise. MCC can be formulated as

$$\text{Max} \sum_{l=1}^{L} c^l \lambda^l, \tag{8}$$

$$\sum_{l=1}^{L} y_i^l \lambda^l = 1, \quad i = 1, \ldots, |V|, \tag{9}$$

$$\sum_{l=1}^{L} \lambda^l = K, \tag{10}$$

$$\lambda^l \in \{0, 1\}, \quad l = 1, \ldots, L, \tag{11}$$

where c^l is the cost of the lth cluster, (9) ensures that each node belongs to exactly one cluster, (10) ensures that K clusters are present in the partition, and the presence or absence of a cluster in the solution is modeled by restricting the variables λ^l to be binary. The problem defined by (8)–(11), which is a set partitioning problem with a cardinality constraint, is called the master problem and is denoted by MP. Its linear programming relaxation is denoted by LPMP.

Since the columns in LPMP are restricted to be 0–1 solutions to the knapsack constraint (6), LPMP provides at least as strong a bound on the optimal objective value of MCC as the bound provided by LPMCC. That LPMP might be strong can be shown explicitly for a special case for which there is no particular reason to suspect that LPMP would be strong. Suppose that G is complete and has Kq nodes with K and q positive integers at least 2. Let $c_e=1$, $e\in E$, and suppose that feasibility requires that every cluster has q nodes. Then any partition of the nodes into K feasible clusters is an optimal integer solution with objective value $\frac{1}{2}Kq(q-1)$. We verify by duality that LPMP also has an optimal solution of this value. Note, however, that LPMCC has an optimal solution $x_i^k=z_e^k=1/K$, $i\in V$, $e\in E$, $k=1,\ldots,K$, with objective value $\sum_{e\in E} c_e=\frac{1}{2}Kq(Kq-1)$, which is approximately K times the optimal integer objective function value.

Let π_i, $i=1,\ldots,|V|$ be the dual variables for (9), and let μ be the dual variable for (10). Then the dual of LPMP is

$$\text{Min} \sum_{i\in V} \pi_i+\mu K, \tag{12}$$

$$\sum_{i\in V} y_i^l\pi_i+\mu \geqslant c^l, \quad l=1,\ldots,L. \tag{13}$$

Since $c^l=\frac{1}{2}q(q-1)$, for all $e\in E$, the solution $\pi_i=0$ for $i\in V$, and $\mu=\frac{1}{2}q(q-1)$ is an optimal solution to the dual with the same objective value as the integer solution to MP.

2.2. *Solving LPMP by column generation*

Since LPMP can have a huge number of columns, to list all the columns is impractical if not impossible. Instead, we start with a few columns and solve the restricted LPMP, adding "good" columns as required, until LPMP is optimized over all possible columns. The small number of initial columns in the restricted LPMP may actually provide a good incumbent solution.

For a feasible cluster with node set $U\subseteq V$, let

$$x_i=\begin{cases}1, & i\in U,\\ 0, & \text{otherwise},\end{cases} \qquad z_1=\begin{cases}1, & (i,j)\in E(U),\\ 0, & \text{otherwise}.\end{cases}$$

When the current restricted LPMP has π_i, $i=1,2,\ldots,|V|$, and μ as the dual variables to (9) and (10) respectively, the subproblem that must be optimized to generate the most attractive column (based on its reduced cost) is

$$-\mu K+\text{Max}\left(\sum_{e\in E} c_e z_e-\sum_{i\in V}\pi_i x_i\right), \tag{14}$$

$$z_e\leqslant x_i, \quad z_e\leqslant x_j, \qquad e=(i,j)\in E, \tag{15}$$

$$\sum_{i\in V} w_i x_i\leqslant b, \tag{16}$$

$$z_e\geqslant 0, \quad e\in E, \quad \text{and} \quad x_i\in\{0,1\}, \quad i\in V. \tag{17}$$

We refer to this subproblem as the knapsack quadratic program (KQP). If constraint (16) is ignored then KQP models the Boolean quadratic program. If the objective value of the optimal solution is positive, then a column corresponding to the optimal vector x may be added to the restricted LPMP to enter the basis. Otherwise, there are no more columns to add, and the current solution to LPMP is optimal. If this optimal solution to LPMP is integral, an optimal solution to MP is at hand; otherwise we use branch-and-bound to obtain an integer solution. Note that since KQP is NP-hard, it follows from the equivalence of separation and optimization that the linear program LPMP is NP-hard.

3. Subproblem optimization

We use a cutting plane/branch-and-bound method for solving the subproblem defined by (14)–(17). This is an LP based procedure, where we generate violated inequalities as needed to cut off the current fractional solution. When our procedure is unable to identify any violated inequalities, we use branch-and-bound to determine an optimal solution. Some of the valid inequalities and their strength are discussed next.

3.1. *Subproblem polytope—Valid inequalities*

We denote the convex hull of the set of solutions to (15)–(17) by the polytope $P_{\mathrm{KQP}}(G)$. Here, we discuss some valid inequalities for $P_{\mathrm{KQP}}(G)$. Note that although valid inequalities for the polytope associated with the Boolean quadratic program [16] are valid for $P_{\mathrm{KQP}}(G)$, these inequalities are not strong enough for solving KQP because of the knapsack constraint.

For $C \subseteq V$, we say that C is an *independent set* if $\sum_{i \in C} w_i \leqslant b$; otherwise C is a *dependent set*. A dependent set is *minimal* if all of its subsets are independent. Note that if C is a minimal dependent set, then $\sum_{j \in C} x_j \leqslant |C| - 1$ is a valid inequality for $P_{\mathrm{KQP}}(G)$.

We assume, for simplicity, that for each $e = (i, j) \in E$, $w_i + w_j \leqslant b$. This assumption implies that $P_{\mathrm{KQP}}(G)$ is full dimensional because the following $|V| + |E| + 1$ affinely independent integer solutions are in $P_{\mathrm{KQP}}(G)$.

(a) $x_i = 0, \quad i \in V, \qquad z_{ij} = 0, \quad (i, j) \in E.$

(b) $x_u = 1, \qquad x_i = 0, \quad i \in V \setminus \{u\}, \qquad z_{ij} = 0, \quad (ij) \in E.$

(c) $x_u = x_v = 1, \qquad x_i = 0, \quad i \in V \setminus \{u, v\}, \qquad z_{uv} = 1, \qquad z_e = 0, \quad e \in E \setminus \{(u, v)\}.$

A valid inequality $\pi x \leqslant \pi_0$ defines a facet of a full dimensional polyhedron P if and only if its coefficients are determined uniquely up to scalar multiplication by points in P (Nemhauser and Wolsey [15]). We use this fact in the proofs of the following results.

For $S \subseteq V$, we let $E(S) = \{(u, v) \mid u, v \in S\}$, and for $F \subseteq E$, we let $V(F)$ denote the set of nodes in $G(V, E)$ consisting of end nodes of the edges in F.

Theorem 1. *Let C be a dependent set and suppose that the graph $G'(C, E(C))$ is connected. Let T be a set of edges in G that form a spanning tree on the nodes in C. Let $\delta(i)=\{j: (i,j)\in T\}$ for each $i\in C$. Then the following tree inequality is valid for $P_{\mathrm{KQP}}(G)$.*

$$\sum_{e\in T} z_e \leqslant \sum_{i\in C} (|\delta(i)|-1)x_i. \tag{18}$$

A tree inequality (18) *defines a facet for $P_{\mathrm{KQP}}(G')$ where $G'=(C, E(C))$ if and only if $C\setminus\{i\}$ is an independent set for every leaf i of the tree induced by T.*

Proof. We first establish the validity of (18). Consider an arbitrary feasible solution in $P_{\mathrm{KQP}}(G)$.

$$x_i=\begin{cases}1, & i\in S,\\ 0, & \text{otherwise,}\end{cases} \qquad z_e=\begin{cases}1, & e\in E(S),\\ 0, & \text{otherwise.}\end{cases}$$

The edges in $E(S)\cap T$ induce a forest F. Let $T'\subseteq F$ be the set of edges in one component of the forest and let $V(T')=S'$. Hence T' induces a tree on the node set S'.

For $i\in S'$, let $\delta'(i)=\{j: (i,j)\in T'\}$. Since $|S'|<|C|$ because C is a dependent set, there exists an edge $(i,j)\in T\setminus T'$, $j\in S'$, so that $|\delta(j)|>|\delta'(j)|$. Since $z_e=1$, $e\in T'$, it follows that

$$\begin{aligned}\sum_{e\in T'} z_e &= |T'| = |S'|-1 = 2(|S'|-1)-|S'|+1\\ &= \sum_{i\in S'} |\delta'(i)| - |S'| + 1 = \sum_{i\in S'} (|\delta'(i)|-1)+1\\ &\leqslant \sum_{i\in S'} (|\delta(i)|-1) = \sum_{i\in S'} (|\delta(i)|-1)x_i,\end{aligned}$$

where the last equality follows because $x_i=1$ for $i\in S'$. Adding these inequalities for all the components of F shows that the tree inequality is satisfied by any point in $P_{\mathrm{KQP}}(G)$.

Suppose that $C\setminus\{i\}$ is an independent set for every leaf i of the tree induced by T. Let $\pi z+\sigma x\leqslant\pi_0$ be a facet defining inequality for $P_{\mathrm{KQP}}(G')$ which contains all equality solutions to (18). We will use some integer points in $P_{\mathrm{KQP}}(G')$ that satisfy (18) at equality to determine the coefficients π and σ uniquely up to scalar multiplication. Since $(z, x)=(0, 0)$ is such a feasible solution, $\pi_0=0$.

Next we prove that $\pi_e=0$ for any edge $e=(i,j)$ such that $e\notin T$. For any nontree edge $e=(i,j)$, there exists a unique path from i to j consisting of edges in the set $P\subseteq T$. The length of a path is the number of edges on the path. Form a list $L=\{e_1, \ldots, e_m\}$ of nontree edges in nonincreasing order of their path lengths. We prove that $\pi_{e_t}=0$, $t=1, \ldots, m$, by induction on t.

Given $i\in V(T)$, let F_i be the forest obtained by deleting the edges (i,j), $j\in\delta(i)$. We denote the $|\delta(i)|$ components of F_i by T_i^j, $j\in\delta(i)$, where T_i^j is the tree that corresponds

to the component that contains node j. Let $T_i(j) = T_i^j \cup \{(i,j)\}$, $K_i^j = T \setminus T_i(j)$, and

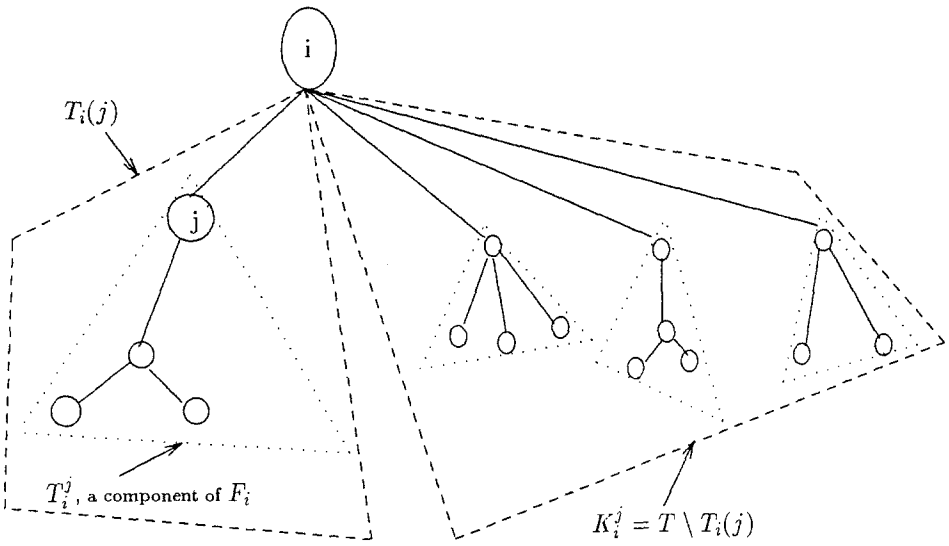

Fig. 1.

let $V_i^j = V(K_i^j) \cup \{i\}$ (see Figure 1). Note that $V_i^j = V(K_i^j)$ if i is not a leaf of the tree induced by T.

Let $e_1 = (i_1, j_1)$ and let P be the set of nodes on the unique path from i_1 to j_1 consisting of edges in T. Let $p = \delta(i_1) \cap P$, and let $q = \delta(j_1) \cap P$. Note that there is no nontree edge (except edge e_1) between the nodes in $V_{i_1}^p$ and the nodes in $V_{j_1}^q$ because the path length of such an edge would be strictly greater than the path length of e_1, which is impossible, since e_1 is the first element in L.

The following solutions satisfy (18) at equality:

$$\text{(a)} \quad x_i = \begin{cases} 1, & i \in V_{i_1}^p, \\ 0, & \text{otherwise}, \end{cases} \qquad z_e = \begin{cases} 1, & e \in E(V_{i_1}^p), \\ 0, & \text{otherwise}, \end{cases}$$

$$\text{(b)} \quad x_i = \begin{cases} 1, & i \in V_{j_1}^q, \\ 0, & \text{otherwise}, \end{cases} \qquad z_e = \begin{cases} 1, & e \in E(V_{j_1}^q), \\ 0, & \text{otherwise}, \end{cases}$$

$$\text{(c)} \quad x_i = \begin{cases} 1, & i \in V_{i_1}^p \cup V_{j_1}^q, \\ 0, & \text{otherwise}, \end{cases} \qquad z_e = \begin{cases} 1, & e \in E(V_{i_1}^p \cup V_{j_1}^q), \\ 0, & \text{otherwise}. \end{cases}$$

We establish this only for (a). Let $|V_{i_1}^p| = m$. The number of edges in $K_{i_1}^p$ is $m-1$, and from the observation made at the beginning of the proof, $m-2 = \sum_{k \in V_{i_1}^p} (|\delta'(k)| - 1)$ where $\delta'(k) = \{j: (k,j) \in K_{i_1}^p\}$. Note that $|\delta(k)| = |\delta'(k)|$ for $k \in V_{i_1}^p \setminus \{i_1\}$ and $|\delta(i_1)| = |\delta'(i_1)| + 1$. Hence $m-1 = \sum_{k \in V_{i_1}^p} (|\delta(k)| - 1)$, and this establishes that the solution consisting of nodes and edges in $K_{i_1}^p$ satisfies (18) at equality.

From the solutions (a)–(c), we get the following equalities.

$$\pi z^{K^p_{i_1}} + \sigma x^{V^p_{i_1}} = 0,$$

$$\pi z^{K^q_{j_1}} + \sigma x^{V^q_{j_1}} = 0,$$

$$\pi z^{K^p_{i_1}} + \pi z^{K^q_{j_1}} + \sigma x^{V^p_{i_1}} + \sigma x^{V^q_{j_1}} + \pi_{e_1} = 0.$$

Subtracting the left hand sides of the first two equations from the left hand side of the third gives $\pi_{e_1} = 0$.

For the induction, assume that $\pi_{e_t} = 0$, $t = 1, \ldots, l-1$. Let $e_l = (i_l, j_l)$ and let P be the set of tree edges on the unique path from i_l to j_l. Let $p = \delta(i_l) \cap P$, and let $q = \delta(j_l) \cap P$. Let the set of nontree edges that have one end node in $V^p_{i_l}$ and the other end node in $V^q_{j_l}$ be denoted by NT.

Using the above arguments, we see that the following solutions satisfy (18) at equality:

(d) $$x_i = \begin{cases} 1, & i \in V^p_{i_l}, \\ 0, & \text{otherwise}, \end{cases} \qquad z_e = \begin{cases} 1, & e \in E(V^p_{i_l}), \\ 0, & \text{otherwise}, \end{cases}$$

(e) $$x_i = \begin{cases} 1, & i \in V^q_{j_l}, \\ 0, & \text{otherwise}, \end{cases} \qquad z_e = \begin{cases} 1, & e \in E(V^q_{j_l}), \\ 0, & \text{otherwise}, \end{cases}$$

(f) $$x_i = \begin{cases} 1, & i \in V^p_{i_l} \cup V^q_{j_l}, \\ 0, & \text{otherwise}, \end{cases} \qquad z_e = \begin{cases} 1, & e \in E(V^p_{i_l} \cup V^q_{j_l}), \\ 0, & \text{otherwise}, \end{cases}$$

From the solutions (d)–(f), we get the following equalities.

$$\pi z^{E(V^p_{i_l})} + \sigma x^{V^p_{i_l}} = 0,$$

$$\pi z^{E(V^q_{j_l})} + \sigma x^{V^q_{j_l}} = 0,$$

$$\pi z^{E(V^p_{i_l})} + \pi z^{E(V^q_{j_l})} + \sigma x^{V^p_{i_l}} + \sigma x^{V^q_{j_l}} + \pi_{e_l} + \sum_{e \in \mathrm{NT} \setminus \{e_l\}} \pi_e = 0.$$

Note that $\pi_e = 0$ for $e \in \mathrm{NT} \setminus \{e_l\}$ by the induction hypothesis. Subtracting the left hand side of the first two equations from the left hand side of the third gives $\pi_{e_l} = 0$. Thus, $\pi_e = 0$ for any nontree edge e.

If i is a leaf of the tree induced by T, then the solution $x_i = 1$, $x_j = 0$, $j \neq i$, $z_e = 0$ for all e, satisfies (18) at equality. Hence $\sigma_i = 0$ for every i such that i is a leaf of the tree induced by T. Next we establish the coefficients σ_i for $i \in C$, such that $|\delta(i)| \geq 2$. Consider the following solutions:

$$x_k = \begin{cases} 1, & k \in V^j_i, \\ 0, & \text{otherwise}, \end{cases} \qquad z_e = \begin{cases} 1, & e \in E(V^j_i), \\ 0, & \text{otherwise}, \end{cases} \qquad j = 1, \ldots, |\delta(i)|.$$

The feasibility of these solutions follows from the assumption that $c \setminus \{k\}^*$ is an independent set whenever k is a leaf of the tree induced by T. Using the previous arguments, it is easy to verify that these solutions satisfy (18) at equality. From these solutions, we get the following equalities.

$$\pi z^{K_i^j} + \sigma x^{V_i^j \setminus \{i\}} + \sigma_i = 0, \quad j = 1, \ldots, |\delta(i)|.$$

These equalities yield

$$\pi z^{T_i(j_1)} + \sigma x^{V(T_i(j_1)) \setminus \{i\}} = \pi z^{T_i(j_2)} + \sigma x^{V(T_i(j_2)) \setminus \{i\}}$$
$$= \gamma_i, \quad j_1, j_2 \in \{1, \ldots, |\delta(i)|\},$$

where γ_i is a constant for node i. Substituting this into the equality corresponding to any one of the $|\delta(i)|$ solutions establishes that

$$\sigma_i + (|\delta(i)| - 1)\gamma_i = 0.$$

We now establish that $\pi_{ij} = \gamma_i$ for $\{(i,j) : j \in \delta(i)\}$. Since $T_i(j) = K_j^i \cup \{(i,j)\}$,

$$\gamma_i = \pi z^{K_j^i} + \sigma x^{V_j^i} + \pi_{ij}.$$

This establishes that $\pi_{ij} = \gamma_i$ because

$$\pi z^{K_j^i} + \sigma x^{V_j^i} = \pi_0 = 0.$$

Repeating the same argument for every tree edge incident to node i, we get $\pi_{ij} = \gamma_i$ for every $j \in \delta(i)$.

If $e = (i,j) \in T$, then the above arguments imply that $\gamma_i = \pi_e = \gamma_j = \gamma$. This establishes all the coefficients uniquely up to scalar multiplication and proves that (18) defines a facet for $P_{\mathrm{KQP}}(G')$.

Now suppose that there exists a node $u \in C$ such that u is a leaf of the tree induced by T, and $C' = C \setminus \{u\}$ is a dependent set. Let $(u, v) \in T$, and $T' = T \setminus \{(u, v)\}$. Let $\delta'(i) = \{j : (i,j) \in T'\}$ for each $i \in C'$. Then the following inequality is valid for $P_{\mathrm{KQP}}(G)$.

$$\sum_{e \in T'} z_e \leqslant \sum_{i \in C'} (|\delta'(i)| - 1)x_i. \tag{19}$$

Since $|\delta'(i)| = |\delta(i)|$, for $i \in C \setminus \{u, v\}$, $|\delta'(v)| = |\delta(v)| - 1$, and $|\delta(u)| = 0$, inequality (18) is a sum of $z_{uv} \leqslant x_u$ and the inequality (19). Hence (18) is not facet defining for $P_{\mathrm{KQP}}(G')$. □

When the tree in inequality (18) is a star, we get the following result.

Corollary 2. *Let C be a minimal dependent set. Then the following star inequalities are valid for $P_{\mathrm{KQP}}(G)$ and are facet defining for $P_{\mathrm{KQP}}(G')$ where $G' = (C, E(C))$.*

$$\sum_{j \in C \setminus \{i\}} z_{ij} \leqslant (|C| - 2)x_i, \quad i \in C. \qquad \square \tag{20}$$

Theorem 1 can be generalized to consider forests rather than just spanning trees over the nodes of C.

Theorem 3. *Let C be a minimal dependent set and let F be a set of edges in G that form a forest on the nodes in C. Let F have p components, and let $\delta(i)=\{j:(i,j)\in F\}$ for each $i\in C$. Then the following forest inequality is valid for $P_{\mathrm{KQP}}(G)$.*

$$\sum_{e\in F} z_e \leqslant (p-1)+\sum_{i\in C}(|\delta(i)|-1)x_i. \tag{21}$$

The inequality is facet defining for $P_{\mathrm{KQP}}(G')$ where $G'=(C, E(C))$ if and only if there are no edges in $E(C)$ between nodes of C in different components of F and $C\setminus\{i\}$ is an independent set for every i such that $|\delta(i)|\leqslant 1$.

Proof. Let T be a set of edges that include all the edges in F and some artificial edges not in $G(V, E)$ that form a spanning tree on the nodes in C. Then adding the tree inequality (18), and the inequalities $z_{ij}\geqslant x_i+x_j-1$ for all the edges in $T\setminus F$ gives the forest inequality (21). Hence, the validity of the forest inequality follows from the validity of the tree inequality.

Now suppose that there are no edges in $E(C)$ between nodes of C in different components of F and $C\setminus\{i\}$ is an independent set for every i such that $|\delta(i)|\leqslant 1$. Let $\pi z+\sigma x\leqslant\pi_0$ be a facet defining inequality for $P_{\mathrm{KQP}}(G)$ which contains all the equality solutions to (21). Let $T_1,\ldots, T_p$ be the trees that correspond to the p components of F. Let V_{T_i} be the set of nodes in component i. Hence, unless $T_i=\emptyset$, $V_{T_i}=V(T_i)$.

Using arguments similar to the arguments used in the proof of Theorem 1, it is easy to see that the solutions

$$x_j=\begin{cases}1, & j\in C\setminus V_{T_i},\\ 0, & \text{otherwise},\end{cases}\qquad z_e=\begin{cases}1, & e\in E(C\setminus V_{T_i}),\\ 0, & \text{otherwise},\end{cases}$$

$i=1,\ldots,p$, satisfy (21) at equality. Let

$$\gamma_i=\pi z^{E(V_{T_i})}+\sigma x^{V_{T_i}}.$$

This implies that

$$\sum_{j=1}^{p}\gamma_j-\gamma_i=\pi_0,\quad i=1,\ldots,p.$$

These equalities give

$$\gamma_i=\gamma_j=\gamma,\quad i,j\in\{1,\ldots,p\},\qquad \gamma=\pi_0/(p-1).$$

We now establish the coefficients of the edge and node variables that correspond to T_1. We consider only those solutions that have all the edges in $T_2\cup\cdots\cup T_p$. If we subtract the sum $\sum_{j=2}^{p}\gamma_j$ from the forest inequality corresponding to such solutions, our analysis reduces to considering solutions that satisfy

$$\sum_{e\in T_1} z_e\leqslant\sum_{i\in V_{T_1}}(|\delta(i)|-1)x_i$$

at equality. Note that at most $|T_1|-1$ nodes can be present in any feasible solution that already contains the edges in $T_2\cup\cdots\cup T_p$. Hence, an inequality of the same form as the tree inequality (18) holds for the nodes in V_{T_1}, and the edges in T_1. The

coefficients can be uniquely determined (up to scalar multiplication) by arguments analogous to those used to establish the coefficients in (18). □

3.2. Lifting

We now discuss the lifting procedure for the star and tree inequalities. Since the star inequalities (20) and tree inequalities (18) were shown to be facet defining for $P_{\mathrm{KQP}}(G')$, where $G'=(C, E(C))$, the variables $x_i \in C$, and z_e, $e \in E(C)$, can not be lifted. We discuss lifting the other variables next.

First consider lifting star inequalities (20). It is easy to verify that in any sequential lifting procedure where maximum lifting is done, the lifting coefficient is 0 for the following variables

(i) z_{pq}, $p, q \notin C$;
(ii) x_p, $p \in V \setminus C$; and
(iii) z_{pj}, $j \in C$, $j \neq i$, $p \notin C$.

Now consider lifting variables z_{ij}, $j \in V \setminus C$. Let $S \subseteq V \setminus C$, and suppose that the following inequality is obtained after maximum lifting of variables z_{ij}, $j \in S$.

$$\sum_{j \in S} \alpha_j z_{ij} + \sum_{j \in C} z_{ij} \leqslant (|C|-2)x_i, \quad i \in C,\ S \subseteq V \setminus C. \tag{22}$$

Consider lifting $y = z_{ip}$, $p \notin \{C \cup S\}$, and let

$$\xi = \max \left\{ \sum_{j \in S} \alpha_j z_{ij} + \sum_{j \in C} z_{ij} - (|C|-2)x_i,\ (z, x) \in P_{\mathrm{KQP}}(G) \cap \{z_{ip} = 1\} \right\}.$$

Note that $z_{ip} = 1 \Rightarrow x_i = x_p = 1 \Rightarrow z_{ij} = 1$ if and only if $x_j = 1$ for any $j \in C$, $j \neq i$. Hence the lifting coefficient for an edge variable z_{ip} is $\gamma \leqslant -\xi$, where

$$\xi = \max \sum_{j \in S} \alpha_j x_j + \sum_{j \in C \setminus \{i\}} x_j - (|C|-2), \qquad \sum_{j \in S \cup C \setminus \{i\}} w_j x_j \leqslant b - w_i - w_p.$$

This is a 0–1 knapsack problem and can be solved by dynamic programming, see [15]. Note that this problem can be solved by a simple sorting of the weights when all the objective coefficients are 1, as is the case when the first edge variable is lifted.

Now consider lifting the tree inequalities (18). It is easy to verify that in any sequential lifting procedure where maximum lifting is done, the lifting coefficient is 0 for the following variables.

(i) z_{pq}, $p, q \notin C$;
(ii) x_j, $j \notin C$; and
(iii) z_{pj}, j a leaf of the tree induced by T, $p \notin C$.

The lifting problem for an edge variable z_{ij}, $i \in C$, $j \in V \setminus C$, where i is not a leaf of the tree induced by T, is a KQP. However, when no edge variable has been lifted

with a positive coefficient, it reduces to a knapsack problem which can be solved by sorting (see [13]).

3.3. Separation

The separation problems for the inequalities discussed in the previous section involve finding a minimal dependent set that minimizes a linear function. Since this problem is NP-hard, see [15], it follows that the separation problem for our inequalities is NP-hard. Therefore, we suggest heuristic procedures to find the violated inequalities.

Let (z^*, x^*) be a fractional solution of the LP relaxation of KQP. Then the separation problem for finding a violated star inequality corresponding to a node $i \in V$ involves finding a set $C \subseteq V$, $i \in C$ (assuming one exists) with

$$\sum_{j \in C} w_j > b \quad \text{and} \quad \sum_{j \in C \setminus \{i\}} z_{ij}^* > (|C| - 2)x_i^* .$$

Let $N(i) = \{k: (i, k) \in E\}$. Introducing a binary vector y to represent the unknown set C, we attempt to choose y such that

$$\sum_{j \in N(i)} w_j y_j > b - w_i ,$$

and $\sum_{j \in N(i)} (x_i^* - z_{ij}^*) y_j$ is minimized. This is a knapsack problem and can be solved approximately using a greedy heuristic, see [15]. The corresponding inequality can then be lifted.

To find a violated forest inequality, we first use a greedy approach to find a violated tree inequality. The greedy algorithm starts with an edge corresponding to the biggest fractional value. At each subsequent iteration, out of all the edges with one end node in the tree and the other end node not in the tree, the edge corresponding to the minimum value of $x_i - z_{ij}$, with z_{ij} fractional, where i is in the tree and j is not in the tree, is added to the tree. The process stops as soon as the nodes in the tree correspond to a dependent set or when there is no fractional z_{ij} such that node i is in the tree and node j is not in the tree. In the latter case, the tree obtained so far represents one component of the forest and other components are found similarly until a forest is obtained such that the nodes in the forest correspond to a dependent set or there are no more fractional edges to consider.

3.4. Decomposition

Here, we discuss a decomposition based method to solve the subproblem defined by (14)–(17). A set S_p of nodes is feasible if $\sum_{i \in S_p} w_i \leqslant b$. For each feasible set S_p, $p = 1, \ldots, P$, define the incidence vector y^p where $y_i^p = 1$ if node i belongs to S_p, and

$y_i^p = 0$ otherwise. The resulting LP relaxation of the (sub)master problem is of the form

$$\text{Max} \sum_{e \in E} c_e z_e - \sum_{i,p} (\pi_i y_i^p) \gamma^p, \tag{23}$$

$$z_{ij} - \sum_p y_i^p \gamma^p \leqslant 0, \quad z_{ij} - \sum_p y_j^p \gamma^p \leqslant 0, \qquad (i,j) \in E, \tag{24}$$

$$\sum_{p=1}^{P} \gamma^p = 1, \tag{25}$$

$$\gamma^p \geqslant 0, \quad p = 1, \ldots, P. \tag{26}$$

The (sub)master problem is started with a few columns. When v_{ij}, v_{ji}, and θ are the dual variables to (24)–(25), the (sub)subproblem to generate a column that can enter the basis of the (sub)master problem is of the form

$$-\theta + \text{Max} \sum_{i \in V} \left(\sum_{(i,j) \in E} (v_{ij} + v_{ji}) - \pi \right) x_i, \tag{27}$$

$$\sum_{i \in V} w_i x_i \leqslant b, \tag{28}$$

$$x_i \in \{0, 1\}, \quad i \in V. \tag{29}$$

The relative strength of the (sub)master program depends on how much of the difficulty is in satisfying the knapsack constraint and how much is in the nature of the costs. For example, where the knapsack constraint is just a cardinality constraint, that constraint is easy to satisfy and decomposition/column generation may not be needed. The advantage of this approach is that the columns generated by the (sub)subproblem may be kept in a bank to be priced out when needed.

4. Computational results

Our attempts to solve real compiler construction problems using the formulation in Section 1 with branch-and-bound were unsuccessful. However, we were able to solve these problems using the decomposition and column generation based methodology. The results we obtained yielded significant improvements over the heuristics used previously. In all the results presented here, the initial restricted LPMP has columns corresponding to an identity solution (each node is in a cluster by itself), and more columns are generated by solving the subproblem to optimality until LPMP is optimized. When the optimal solution to LPMP is fractional, a simple branch-and-bound methodology is used to determine an integer solution without generating more columns. This can of course yield a suboptimal solution. In the 12 problems we solved, only 2 master problems had fractional optimal solutions and in both of them, the integer solution obtained by branch-and-bound was very close to the LP solution in objective value.

Table 1
Graph description

Graph number	Connected nodes	Number of edges
1	45	98
2	30	56
3	47	101
4	47	99
5	30	47
6	61	187

The optimal solution to the subproblem can sometimes correspond to nodes that do not induce a connected graph. In that case, each component of the induced subgraph is added as a separate column provided that it prices out positively.

All programs are in FORTRAN using IBM's Optimization Subroutine Library and the timings reported are on the RS 6000 (Model 540). In all the runs, we first identify the isolated nodes which are placed in clusters by themselves. This identification of isolated nodes is a simple preprocessing step to reduce the size of the problem. Table 1 describes the sizes of the graphs after this reduction. Tables 2 and 3 summarize the results. In the first set $b = 521K$ and in the second set $b = 450K$.

Table 2
Summary of results ($b = 512$)

Graph number	CPU sec	Number of columns	LP value	IP value
1	935.92	72	3238	3238
2	102.41	36	1748	1748
3	843.53	97	3969	3969
4	2549.52	92	1993	1993
5	95.40	48	1174	1174
6	3962.64	153	23564	23564

Table 3
Summary of results ($b = 450$)

Graph number	CPU sec	Number of columns	LP value	IP value
1	570.42	64	2928	2928
2	53.59	30	1642	1642
3	898.42	92	3574	3569
4	1732.40	67	1837	1837
5	60.46	27	1099	1099
6	3277.17	128	22245	22142

Next, we discuss our results on subproblem optimization for column generation. We implemented a cutting plane/branch-and-bound procedure for solving the subproblems. Adding violated lifted star inequalities in a cutting plane procedure before using branch-and-bound was effective for solving the subproblems.

We demonstrate the strength of the star inequalities on 12 subproblems that appeared in the column generation procedure. The problems selected reflect the level of difficulty of the subproblems at various stages of the column generation procedure. These subproblems are for the problem corresponding to graph 2. In Table 4 the problem code lists the right hand side value of the knapsack constraint and the

Table 4
Subproblem results

Problem code	LP value	Lifted star inequality	MIP value	% gap reduction
450-2	867.28	731.99	728.0	97.1
450-11	361.02	123.58	102.0	91.7
450-16	315.51	100.95	55.0	82.3
450-28	312.45	97.48	10.0	71.0
512-2	1108.28	956.53	887.0	68.6
512-11	555.56	510.69	503.0	85.4
512-21	321.22	122.88	101.0	90.0
512-31	305.62	112.73	25.4	68.8
512-5	934.60	688.87	640.0	83.4
512-15	354.80	193.65	163.0	84.0
512-25	309.58	112.89	35.6	71.8
512-35	306.48	110.32	6.6	65.4

column number for which this subproblem is solved. Table 4 also lists the objective values of the LP relaxation, the value after adding some lifted star inequalities, and the MIP objective value. The last column lists the percentage reduction of the gap between the LP objective value and the MIP objective value by addition of lifted star inequalities.

The separation method in our implementation for a star inequality corresponding to node i is a greedy algorithm that adds edges $(i,j), j \in N(i)$, with a high fractional value until a dependent set is obtained. This dependent set is then reduced to a minimal dependent set and the corresponding star inequality is lifted by using dynamic programming. As shown in Section 3.3, the separation problem for the star inequalities is a knapsack problem. We experimented using dynamic programming to solve this knapsack problem to determine the initial dependent set. The results obtained are similar to the case when the greedy algorithm is used.

We also implemented a heuristic separation algorithm and an exact lifting procedure for tree inequalities (18). The improvement in the results obtained on adding violated lifted tree inequalities after addition of star inequalities was negligible.

Finally, we implemented the decomposition algorithm described in Section 3.4 for solving the subproblems; the results were not competitive with those obtained by a cutting plane/branch-and-bound method.

4.1. Issues in column generation

There are different stages of column generation in the decomposition scheme and, for the sake of efficiency, column generation procedures can be tailored to these different stages.

The initial restricted LPMP can be started with a few columns providing a starting basis. However, if the corresponding dual solution is very degenerate, several iterations of column generation may be required before any columns that appear in the optimal solution are added. An alternative is to include more starting columns so that the dual solution to the LP reflects a later stage of restricted LPMP solution. This can improve the quality of the additional columns generated, at the expense of increasing the size of the restricted LPMP that is solved.

Later in the procedure, the columns generated should price out appropriately to enter the basis to the current restricted LPMP. Note that it is not necessary to generate the most attractive column (based on its reduced cost). Any column that is eligible to enter the current basis may be generated to add to the master program. In fact, several columns may be generated at once before updating the restricted LPMP solution. One method of generating several columns quickly is to do *subset column generation*, that is, partition the nodes into a few subsets, treat the underlying graphs as defining individual clustering problems, and optimize over these separately. Any scheme may be used to do this, however, the decomposition scheme may be the best since this would also provide columns for the overall clustering problem. If the subsets are small enough, it may be practical to generate all possible columns over them. Then only attractive ones may be included in the restricted LPMP and others kept for future use. Efforts to generate these columns individually later may be far more costly than maintaining a bank of these columns for pricing out. If the subsets chosen are nonoverlapping, then they will also provide an incumbent integer solution rather quickly.

The compiler problems did not require implementation of efficient column generation schemes for different stages of the solution procedure. We feel, however, that these issues will be critical in the solution procedure for the more difficult max-cut and mixed-cut clustering problems.

4.2. Distinguished nodes

In certain clustering problems, one or more nodes are specified to be in a given cluster. Even when such a requirement is not explicitly stated, it is possible to have several nodes, no two of which can occur together in any cluster because of the knapsack constraint that limits the sum of the node weights in any cluster. It may then be advantageous to determine these sets of *distinguished nodes* because then the column generation can be done over smaller graphs. We describe a method to identify such distinguished nodes when the number of clusters to be formed is not fixed.

Let the weight of a path p from node i to node j in G be the sum of the weights on all the nodes in the path. We will denote this path weight by $W_p(i,j)$. Let $W_{\min}(i,j) = \min_p\{W_p(i,j)\}$. From the initial graph $G(V,E)$, construct a graph $G'(V,E')$ as follows. Let $(i,j) \in E'$ if and only if $W_{\min}(i,j) > b$. Note that an $(i,j) \in G'$ implies that there exists an optimal solution in which i and j are not in the same cluster. Hence, there exists an optimal solution such that a clique in G' represents a set of nodes no two of which are in the same cluster.

We implemented this idea of distinguished nodes in column generation by solving a subproblem for each distinguished node. Even though each subproblem was easier to solve, the time for solving several subproblems instead of one for each iteration of column generation was substantial. The results obtained using this implementation did not improve the previous ones.

Acknowledgements

We thank Dr. T.K. Phillips of the IBM Research Center for providing the data for the compiler design problems. We also thank the anonymous referees for their careful reading and helpful suggestions. A. Mehrotra gratefully acknowledges the support from an IBM graduate fellowship for 1990–92, and the support received at IBM T.J. Watson Research Center during the summers of 1989–91.

References

[1] F. Barahona and R. Majhoub, "On the cut polytope," *Mathematical Programming* 36 (1986) 157–173.

[2] S. Chopra, "The graph partitioning polytope on series-parallel and 4-wheel free graphs," J.L. Kellogg Graduate School of Management, Northwestern University (Evanston, IL, 1991).

[3] S. Chopra and M.R. Rao, "The partition problem," J.L. Kellogg Graduate School of Management, Northwestern University (Evanston, IL, 1990).

[4] S. Chopra and M.R. Rao, "Facets of the k-partition polytope," J.L. Kellogg Graduate School of Management, Northwestern University (Evanston, IL, 1991).

[5] M. Conforti, M.R. Rao and A. Sassano, "The equipartition polytope. I: Formulations, dimension and basic facets," *Mathematical Programming* 49 (1990) 49–70.

[6] M. Conforti, M.R. Rao and A. Sassano, "The equipartition polytope. II: Valid inequalities, and facets," *Mathematical Programming* 49 (1990) 71-90.

[7] E. Dahlaus, D.S. Johnson, C.H. Papadimitriou, P. Seymour and M. Yanakakis, "The complexity of multiway cuts," extended abstract (1983).

[8] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, New York, 1979).

[9] O. Goldschmidt and D.S. Hochbaum, "An $O(|V|^2)$ algorithm for the k-cut problem," *Proceedings 29th Annual Symposium on Foundations of Computer Science* (1985) pp. 444–451.

[10] M. Grötschel and Y. Wakabayashi, "A cutting plane algorithm for a clustering problem," *Mathematical Programming (Series B)* 45 (1989) 59–96.

[11] M. Grötschel and Y. Wakabayashi, "Facets of the clique partitioning polytope," *Mathematical Programming* 47 (1990) 367–387.

[12] E.L. Johnson, "Modeling and strong linear programs for mixed integer programming," *NATO ASI Series, F51, Algorithms and Model Formulations in Mathematical Programming* (Springer, Berlin, 1989).

[13] A. Mehrotra, "Constrained graph partitioning: decomposition, polyhedral structure and algorithms," PhD Dissertation, School of Industrial and Systems Engineering, Georgia Institute of Technology (Atlanta, GA, 1992).

[14] G.L. Nemhauser and S. Park, "A polyhedral approach to edge coloring," *Operations Research Letters* 10 (1991) 315–322.

[15] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization* (Wiley, New York, 1988).

[16] M. Padberg, "The Boolean quadratic polytope: Some characteristics, facets and relatives," *Mathematical Programming (Series B)* 45 (1989) 139–172.