New Facets and a Branch-and-Cut Algorithm for the Weighted Clique Problem

Michael M. Sørensen
Dept. of Management Science and Logistics
The Aarhus School of Business
Fuglesangs Allé 4
DK-8210 Aarhus V
Denmark

July 19, 2001

Abstract

We consider a polyhedral approach to the weighted maximal b-clique problem. Given a node- and edge-weighted complete graph the problem is to find a complete subgraph (clique) with no more than b nodes such that the sum of the weights of all nodes and edges in the clique is maximal. We introduce four new classes of facet defining inequalities for the associated b-clique polytope. One of these inequality classes constitutes a generalization of the well known tree inequalities; the other classes are associated with multistars. We utilize these inequalities together with other classes of facet defining inequalities in a branch-and-cut algorithm for the problem. We give a description of this algorithm, including some separation procedures, and present the computational results for different sets of test problems. The computation times that are obtained indicate that this algorithm is more efficient than previously described algorithms for the problem.

Keywords: Combinatorial Optimization; Boolean quadratic problem; Edge-weighted clique problem; Maximum dispersion problem; Polyhedra.

1 Introduction

The weighted maximal b-clique problem (WCP) consists in finding a maximum weight clique with no more than b nodes in a node- and edge-weighted complete graph. The weight of the clique is the sum of the weights of all its nodes and edges. In the absence of the node weights this problem is also known as the edge-weighted clique problem. In this paper we pursue a polyhedral approach to the problem which has been founded by other researchers. The results we describe are of two kinds: we introduce new classes of facet defining inequalities for the associated b-clique polytope, and we present a branch-and-cut algorithm that performs better than previously considered exact algorithms for the problem.

The WCP generalizes the Boolean quadratic programming and maximal clique problems. So it is an NP-hard problem. Hunting et al. [5] give references to several applications of the WCP, for example to facility location and -dispersion problems. The problem also arises as the subproblem in a column generation approach to graph partitioning.

The first attempt to solve the edge-weighted version of the WCP by an exact algorithm is described in Dijkhuizen and Faigle [1]. They take a cutting-plane approach which is based on a "natural" edge variable formulation of the problem, but they are only able to solve small problem instances. Subsequently different formulations of the closely related problem of finding a maximal weight clique with exactly b nodes are examined by Faigle et al. [2]. They conclude that a formulation which also includes node variables is much tighter. In a contemporary work on the edge-weighted version of the WCP Park et al. [11] draw the same conclusion. Furthermore, by utilizing results of Johnson et al. [6] and Padberg [10] they provide the first polyhedral results for the b-clique polytope. They are also successful in solving problem instances involving graphs with up to 30 nodes by a cut-and-branch algorithm. Macambira and de Souza [8] follow this line of research by introducing further classes of facet defining inequalities. They examine different cutting strategies for a branch-and-cut algorithm and manage to solve problem instances with up to 48 nodes. Hunting [4] and Hunting et al. [5] also provide several new classes of facet defining inequalities for the b-clique polytope. They utilize some of the inequality classes in a Lagrangian relax-and-cut algorithm by which they solve the same problem instances as in [8].

In all the above studies a "complete graph formulation" of the WCP is used. Mehrotra [9] considers the cardinality constrained Boolean quadratic polytope which is the same as the b-clique polytope, except that edge variables are only

defined for edges of the graph with nonzero weights. Using branch-and-cut he is able to solve problem instances that are associated with a graph on 30 nodes.

Below we define some notation that is used in this paper. Then, in section 2 we state the WCP as a mixed integer linear programming problem using the complete graph formulation and summarize some polyhedral results for the b-clique polytope. We also introduce four new classes of inequalities, cut-tree inequalities and three inequality classes that are associated with multistars, and prove that they are facet defining for the b-clique polytope. Section 3 describes the branch-and-cut algorithm we have designed for the WCP. This algorithm utilizes the new classes of facet defining inequalities together with α- and clique inequalities, and some of the separation procedures it employs are explained. Finally, in section 4 we present the computational results that have been obtained by the algorithm. The problem instances that are solved range in size from 30 to 61 nodes.

We consider the complete graph $K_n = (V_n, E_n)$ on n nodes with node set V_n and edge set E_n. For any subset of nodes $S \subseteq V_n$ we denote by $E_n(S)$ the set of all edges with both endnodes in S. So the subgraph $(S, E_n(S))$ is a clique in K_n. For two disjoint subsets of nodes $S, T \subset V_n$ we define $\delta(S, T)$ to be the set of edges with one endnode in S and the other endnode in T. In the special case where $S = \{v\}$ and $T = V_n \setminus \{v\}$ we use the shorthand notation $\delta(v)$ to refer to the star of node v.

Let $(x, y) \in \mathbb{R}^{n(n+1)/2}$ be a vector with an entry x_v for each node $v \in V_n$ and an entry y_e for each edge $e \in E_n$. Then for every subset $S \subseteq V_n$ we denote by $x(S)$ the sum $\sum_{v \in S} x_v$. Similarly, if σ is a scalar $\sigma x(S)$ denotes $\sum_{v \in S} \sigma x_v$, and for every subset of edges $F \subseteq E_n$ we use $\sigma y(F)$ to denote $\sum_{e \in F} \sigma y_e$ in the obvious way.

Every subset of nodes $C \subseteq V_n$ with $|C| \leq b$ induces a feasible clique in K_n. We use the notation $\left(x^C, y^C\right)$ to refer to the incidence vector of the clique. That is, $x_v^C = 1$ for all $v \in C$, $y_e^C = 1$ for all $e \in E_n(C)$, and $x_v^C = 0$ and $y_e^C = 0$ otherwise.

2 Facets of the b-clique polytope

We denote the b-clique polytope by $P_n(b)$. This is the convex hull of all incidence vectors of cliques in K_n with no more than b nodes. In this section we consider a partial description of $P_n(b)$ in terms of facet defining inequalities. The facets of

$P_n(b)$ are faces $\left\{(x,y) \in P_n(b) \mid a^T(x,y) = a_0\right\}$, defined by valid inequalities $a^T(x,y) \leq a_0$, that are maximal in the sense that they are not contained in any other face of $P_n(b)$. It is well known that the b-clique polytope is full-dimensional. This means that all facets of $P_n(b)$ are unique up to multiplication by a scalar.

We first give a mixed integer LP formulation of the WCP. Given node weights ω_v for all $v \in V_n$ and edge weights c_e for all $e \in E_n$ and a maximal clique size b the weighted b-clique problem can be stated as follows.

$$
\begin{array}{lll}
\max & \displaystyle\sum_{v \in V_n} \omega_v x_v + \sum_{e \in E_n} c_e y_e & \\
\text{s.t.} & & \\
& y_{uv} - x_u \leq 0 \text{ and } y_{uv} - x_v \leq 0 & \text{for all } uv \in E_n \\
& x_u + x_v - y_{uv} \leq 1 & \text{for all } uv \in E_n \\
& y(\delta(v)) - (b-1)x_v \leq 0 & \text{for all } v \in V_n \\
& y_e \geq 0 & \text{for all } e \in E_n \\
& x_v \text{ integer} & \text{for all } v \in V_n.
\end{array}
\tag{1}
$$

The first two sets of constraints guarantee that an edge is in the clique if and only if both of its endnodes are in the clique. The third constraint set, called star inequalities, imposes the size restriction of the clique; without this constraint set we have the Boolean quadratic programming problem considered by Padberg [10]. Although the WCP is stated here as a MILP we remark that (1) implies that all variables are indeed 0–1 variables.

The following proposition states that (1) is a tight formulation of the WCP, because all inequalities define facets of $P_n(b)$ under mild conditions; see Padberg [10] and Park et al. [11] for a proof.

Proposition 1 *Let $n \geq 3$ and $b \geq 2$.*

1. *For every edge $e \in E_n$ the nonnegativity constraint $y_e \geq 0$ defines a facet of $P_n(b)$.*

2. *For every edge $uv \in E_n$ the inequalities $y_{uv} - x_u \leq 0$ and $x_u + x_v - y_{uv} \leq 1$ define facets of $P_n(b)$ if and only if $b \geq 3$.*

3. *For every node $v \in V_n$ the star inequality $y(\delta(v)) - (b-1)x_v \leq 0$ defines a facet of $P_n(b)$ if and only if $b \leq n-1$.* □

In order to obtain a tighter approximation to $P_n(b)$ we shall utilize several other classes of facet defining inequalities. The following class of α-*inequalities* is introduced in Macambira and de Souza [8]. It generalizes the classes of clique-, cut-, and generalized cut inequalities originally obtained by Padberg [10] for the Boolean quadric polytope $P_n(n)$.

Proposition 2 *Let $S, T \subset V_n$ be two disjoint subsets of nodes. For every integer α the α-inequality induced by S and T*

$$\begin{aligned} y(\delta(S,T)) - y(E_n(S)) - y(E_n(T)) \\ - \alpha x(S) + (\alpha - 1)x(T) \leq \alpha(\alpha - 1)/2 \end{aligned} \tag{2}$$

is valid for $P_n(b)$. For every positive integer α the inequality defines a facet of $P_n(b)$ if $|S| \geq 1$, $|T| \geq \alpha + 1$, and

$$b \geq \begin{cases} \alpha + 2, & \text{when } |S| = 1 \\ \alpha + 3, & \text{when } |S| \geq 2. \end{cases}$$

□

The conditions stated in this proposition are not necessary conditions in general. They are necessary conditions with respect to b when $S \neq \emptyset$ and $\alpha = 1$, and in this case the inequality is called a *cut inequality*. When $S = \emptyset$ the α-inequality specializes to a *clique inequality*. In the next proposition we consider this inequality class with coefficient $\beta = \alpha - 1$.

Proposition 3 *For every subset of nodes $T \subseteq V_n$ with $|T| \geq 3$ and every positive integer $\beta \leq |T| - 2$ the clique inequality*

$$\beta x(T) - y(E_n(T)) \leq \beta(\beta + 1)/2 \tag{3}$$

defines a facet of $P_n(b)$ if and only if

$$b \geq \begin{cases} \beta + 2, & \text{when } T \subset V_n \\ \beta + 1, & \text{when } T = V_n. \end{cases}$$

□

The clique inequalities are introduced in Padberg [10] for $P_n(n)$, and except for a minor correction cf. Hunting [4] the above conditions with respect to b are obtained in Park et al. [11].

The next inequalities are described in Hunting et al. [5].

Proposition 4 *For every $v \in V_n$ the inequality*

$$bx_v + x(V_n \setminus \{v\}) - y(\delta(v)) \leq b \tag{4}$$

defines a facet of $P_n(b)$ if and only if $3 \leq b \leq n-2$. □

Hunting [4] also introduces the following inequalities.

Proposition 5 *For all distinct $s, t \in V_n$ the inequality*

$$(b-1)y_{st} - (b-1)x_s + \sum_{v \in V_n \setminus \{s,t\}} (y_{sv} - y_{tv}) \leq 0 \tag{5}$$

defines a facet of $P_n(b)$ if and only if $3 \leq b \leq n-2$. □

In sections 2.2.1 and 2.2.2 we provide new classes of facet defining inequalities that contain inequalities (4) and (5) as special cases.

The *tree inequalities* of the next proposition are introduced in Johnson et al. [6] for the quadratic knapsack polytope; this polytope is defined in Hunting et al. [5] as a generalization of $P_n(b)$. The following closed-form result for $P_n(b)$ is stated in Park et al. [11].

Proposition 6 *Let $G_T = (V_T, E_T) \subset K_n$ be a tree such that $|V_T| = b+1$. Denote by $d_v = |\delta(v) \cap E_T|$ the degree of node v in the tree. Then the tree inequality*

$$y(E_T) - \sum_{v \in V_T} (d_v - 1)x_v \leq 0 \tag{6}$$

defines a facet of $P_n(b)$, for $b \geq 3$, if and only if T is not a star or $V_T = V_n$. □

Further classes of facet defining inequalities for $P_n(b)$ are known. Since, however, we do not utilize any of these inequalities here, we refer the interested reader to Hunting et al. [5], Macambira and de Souza [8], Mehrotra [9], and Sherali et al. [12]. Below we introduce new classes of inequalities that are facet defining for $P_n(b)$.

2.1 Cut-tree inequalities

The first class we consider is a generalization of the tree inequalities (6). This inequality class originates from a study of the closely related simple graph partitioning polytope cf. Sørensen [13]. In that study we have modified tree inequalities by superimposing so-called S,T-inequalities on the tree inequalities. The S,T-inequalities are very similar in structure to the cut inequalities (2), and it turns out that we can modify the tree inequalities (6) by superimposing cut inequalities in an identical manner to obtain a much larger class of inequalities. We call them cut-tree inequalities.

First we need to define some sets of nodes and edges that will be used to describe the cut-tree inequalities. Let $G_T = (V_T, E_T) \subset K_n$ be any tree such that $|V_T| = b + 1$. For each $v \in V_T$ let

$$N_v = \{u \in V_T \setminus \{v\} \mid uv \in E_T\}$$

be the set of neighbor nodes of v in the tree. Let

$$L = \{v \in V_T \mid |N_v| = 1\} \qquad \text{and} \qquad I = V_T \setminus L$$

be the sets of leaf nodes and inner nodes, respectively, of the tree. We shall distinguish between leaf neighbors and inner neighbors of the inner nodes. So for each inner node $i \in I$ denote by

$$L_i = N_i \cap L \qquad \text{and} \qquad I_i = N_i \cap I$$

the two sets of neighbor nodes. Furthermore, we shall also need a so-called neighbor tree for each inner node. For each $i \in I$ let

$T_i \subseteq E_n(I_i)$ be any tree that spans all nodes in I_i.

That is, T_i is a spanning tree of the complete subgraph induced by the inner neighbors of i.

Now we associate additional nodes with the tree. We associate one or more (possibly empty) node sets with each inner node as follows: associate with all $i \in I$ and every $\ell \in L_i$, $uv \in T_i$ mutually disjoint node sets $R_\ell, R_{uv} \subset V_n \setminus V_T$. That is, every pair of these sets has an empty intersection. Given these node sets we define appropriate edge sets:

$$\begin{aligned} &Q_\ell := \delta(\{i\}, R_\ell), \\ &\bar{Q}_\ell := \delta(\{\ell\}, R_\ell) \cup E_n(R_\ell) \qquad \forall \ell \in L_i,\ i \in I, \end{aligned}$$

and

$$Q_{uv} := \delta(\{i\}, R_{uv}),$$
$$\bar{Q}_{uv} := \delta(\{u, v\}, R_{uv}) \cup E_n(R_{uv}) \qquad \forall uv \in T_i,\ i \in I.$$

Then the cut-tree inequality is

$$y(E_T) - \sum_{v \in V_T} (d_v - 1)x_v + \sum_{i \in I} \left(\sum_{\ell \in L_i} ((y(Q_\ell) - y(\bar{Q}_\ell)) + \sum_{uv \in T_i} ((y(Q_{uv}) - y(\bar{Q}_{uv})) \right) \leq 0. \tag{7}$$

This inequality is clearly much more complex in structure than (6). So let us consider an example which will demonstrate the above concepts.

Example Suppose that $b = 8$. Figure 1 shows the support graph of a cut-tree inequality that is obtained in the following way. Consider the tree with edge set

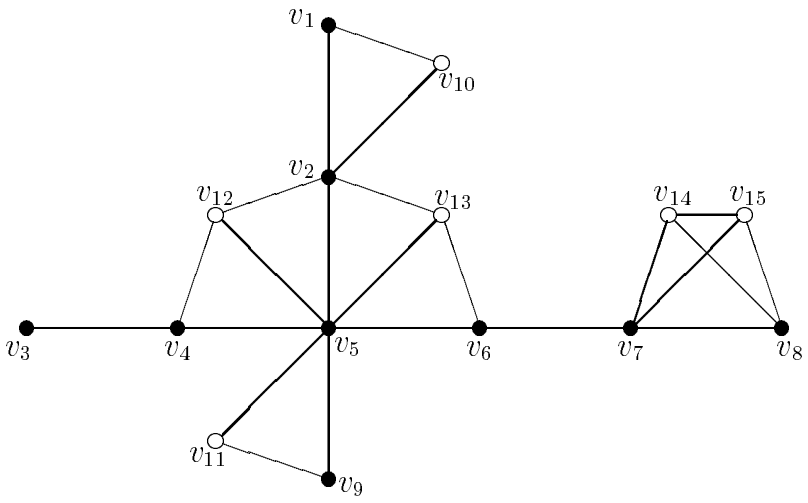

Figure 1: Support graph of a cut-tree inequality.

$$E_T = \{v_1v_2, v_2v_5, v_3v_4, v_4v_5, v_5v_6, v_5v_9, v_6v_7, v_7v_8\},$$

leaf nodes $L = \{v_1, v_3, v_8, v_9\}$ and inner nodes $I = \{v_2, v_4, v_5, v_6, v_7\}$. The neighbor sets are:

$$
\begin{array}{lll}
N_{v_1} = \{v_2\}, & & \\
N_{v_2} = \{v_1, v_5\}, & L_{v_2} = \{v_1\}, & I_{v_2} = \{v_5\}, \\
N_{v_3} = \{v_4\}, & & \\
N_{v_4} = \{v_3, v_5\}, & L_{v_4} = \{v_3\}, & I_{v_4} = \{v_5\}, \\
N_{v_5} = \{v_2, v_4, v_6, v_9\}, & L_{v_5} = \{v_9\}, & I_{v_5} = \{v_2, v_4, v_6\}, \\
N_{v_6} = \{v_5, v_7\}, & L_{v_6} = \emptyset, & I_{v_6} = \{v_5, v_7\} \\
N_{v_7} = \{v_6, v_8\}, & L_{v_7} = \{v_8\}, & I_{v_7} = \{v_6\}, \\
N_{v_8} = \{v_7\}, & & \\
N_{v_9} = \{v_5\}, & &
\end{array}
$$

and a possible choice of neighbor trees is:

$$
\begin{array}{l}
T_i = \emptyset \qquad \text{for all } i \in I \setminus \{v_5, v_6\}, \\
T_{v_5} = \{v_2v_4, v_2v_6\}, \\
T_{v_6} = \{v_5v_7\}.
\end{array}
$$

By associating the following node sets with the leaf neighbors and edges of the neighbor trees of the inner nodes:

$$
\begin{array}{lll}
R_{v_1} = \{v_{10}\}, & & \\
R_{v_3} = \emptyset, & & \\
R_{v_9} = \{v_{11}\}, & R_{v_2v_4} = \{v_{12}\},\ R_{v_2v_6} = \{v_{13}\}, & \\
 & R_{v_5v_7} = \emptyset, \text{ and} & \\
R_{v_8} = \{v_{14}, v_{15}\} & &
\end{array}
$$

we obtain the cut-tree inequality

$$
\begin{aligned}
& y_{v_1v_2} + y_{v_2v_5} + y_{v_3v_4} + y_{v_4v_5} + y_{v_5v_6} + y_{v_5v_9} + y_{v_6v_7} + y_{v_7v_8} \\
& \quad - x_{v_2} - x_{v_4} - 3x_{v_5} - x_{v_6} - x_{v_7} \\
& \quad + y_{v_2v_{10}} - y_{v_1v_{10}} + y_{v_5v_{11}} - y_{v_9v_{11}} \\
& \quad + y_{v_5v_{12}} - y_{v_2v_{12}} - y_{v_4v_{12}} + y_{v_5v_{13}} - y_{v_2v_{13}} - y_{v_6v_{13}} \\
& \quad + y_{v_7v_{14}} + y_{v_7v_{15}} - y_{v_8v_{14}} - y_{v_8v_{15}} - y_{v_{14}v_{15}} \leq 0.
\end{aligned}
$$

□

We have the following result.

Theorem 1 *The cut-tree inequality (7) is valid for $P_n(b)$. It defines a facet of $P_n(b)$, for $b \geq 3$, if and only if the tree is not a star or $V_T = V_n$.*

Proof. We denote the cut-tree inequality by $a^T(x, y) \leq 0$ and first prove validity. Let $C \subset V_n$ with $|C| \leq b$ be the node set of a feasible clique. We consider the intersection of the clique $(C, E_n(C))$ with the support graph of (7) which may consist of several connected components. Let (U, F) be any such connected component. We shall show that $a^T\left(x^U, y^U\right) \leq 0$, because then validity immediately follows by adding these inequalities for all connected components.

In the special case of a tree inequality (6) where all node sets R_ℓ, R_{uv} are empty it follows from arguments similar to those in the proof provided in Johnson et al. [6] that

$$a^T\left(x^U, y^U\right) = \sum_{v \in U'} (|\delta'(v)| - d_v) + 1, \tag{8}$$

where $U' = U \cap V_T$ and $\delta'(v) = \delta(v) \cap F \cap E_T$. Furthermore, $U' \subset V_T$ because $|V_T| > b$, so there exists an edge $st \in E_T \setminus F$, $s \in U'$, such that $|\delta'(s)| < d_s$. Hence, $a^T\left(x^U, y^U\right) \leq 0$ in this case.

In the general case, where some node sets R_ℓ, R_{uv} are nonempty, the right-hand side value of (8) is still nonpositive. So it suffices to show that the right-hand side of (8) plus the contribution from any additional nodes in $U \setminus U'$ is nonpositive. Here we only consider contributions from nodes in the sets R_{uv} that are associated with the edges of the neighbor trees of the inner nodes. The possible contributions from nodes in the sets R_ℓ that are associated with the leaf nodes are easy to establish by similar arguments.

Let $i \in I \cap U$. We consider three cases. *i)* First suppose that $R_{uv} \cap U \neq \emptyset$ for some $uv \in T_i$ and such that $\{u, v\} \cap U \neq \emptyset$. Then, for each node $w \in R_{uv} \cap U$, we get a $+1$ contribution from the edge iw and a -1 contribution from each of the edges uw, vw for which $u, v \in U$. This yields a net contribution of no more than 0 so that $a^T\left(x^U, y^U\right) \leq 0$.

ii) Next suppose that there is one $uv \in T_i$ for which $\{u, v\} \cap U = \emptyset$ and $R_{uv} \cap U \neq \emptyset$. Since $u, v \notin U$, $|\delta'(i)| - d_i \leq -2$. This implies that the right-hand side of (8) is negative. On the other hand we get a $+1$ contribution from the edge iw for each node $w \in R_{uv} \cap U$. Furthermore, if $|R_{uv} \cap U| \geq 2$ we get contributions of -1 for each edge $e \in \bar{Q}_{uv} \cap F$. It follows that $a^T\left(x^U, y^U\right) \leq 0$ in this case, too.

iii) Finally suppose that there are k distinct node sets R_{uv}, $uv \in T_i$, for which $R_{uv} \cap U \neq \emptyset$ and $\{u, v\} \cap U = \emptyset$. We note that $k \leq d_i - 1$. Since T_i spans all inner neighbors of node i, we have $|\delta'(i)| \leq d_i - k - 1$, and it follows that the right-hand side value of (8) is less than or equal to $-k$. Using the arguments

in *ii)* above for each node set R_{uv} it follows that $a^T \left(x^U, y^U\right) \leq 0$. This proves that the cut-tree inequality is valid for $P_n(b)$.

Facet: When G_T is a star there is only one inner node in the tree: $I = \{i\}$. In this case the inequality (7) is dominated by the star inequality $y(\delta(i)) - (b-1)x_i \leq 0$ if $V_T \subset V_n$. This proves the only if statement of the theorem.

In order to prove sufficiency we assume throughout the remainder of this proof that G_T is not a star. Denote by $F_a = \left\{(x, y) \in P_n(b) \mid a^T(x, y) = 0\right\}$ the face of the b-clique polytope that is defined by the cut-tree inequality. Let $\pi^T(x, y) \leq \pi_0$ be a facet defining inequality for $P_n(b)$ such that $F_a \subseteq F_\pi = \left\{(x, y) \in P_n(b) \mid \pi^T(x, y) = \pi_0\right\}$. We shall show that $\pi = \sigma a$ and $\pi_0 = 0$ for some positive scalar $\sigma \in \mathbb{R}_+$, since this implies that $F_a = F_\pi$.

We make extensive use of cliques that are constructed in the following way. Let $i \in I$ and $j \in N_i$ be two adjacent nodes of the tree. By deleting the edge ij from E_T we obtain two subtrees (connected components) of G_T. We denote by $C(i) \subset V_T$ (respectively $C(j) \subset V_T$) the set of nodes of the subtree that contains node i (respectively node j). It is quite easy to see that the incidence vectors of the two corresponding cliques are both members of the face F_a.

(a) First we note that $\left(x^\emptyset, y^\emptyset\right) \in F_a \subseteq F_\pi$ implies $\pi_0 = 0$.

(b) Furthermore, because the tree inequality (6) is facet defining, it follows that $\pi_v = -(d_v - 1)\sigma$ for all $v \in V_T$, $\pi_e = \sigma$ for all $e \in E_T$, and $\pi_e = 0$ for all $e \in E_n(V_T) \setminus E_T$.

(c) Consider any node $s \in R_{uv}$, $uv \in T_i$, or $s \in R_u$, $u \in L_i$, for some $i \in I$. From $\left(x^{\{s\}}, y^{\{s\}}\right) \in F_a$ we get $\pi_s = 0$.

(d) Let i, s, u, v be as in (c), and let $t \in V_T \setminus \{i, u, v\}$. We shall show that $\pi_{st} = 0$. Let $j \in I_t$ be the unique inner neighbor of node t that lies on the path in G_T from t to i and consider the partition $C(j), C(t)$ of V_T. Let $D = C(t) \cup \{s\}$. Then the incidence vectors of $C(t)$ and D are contained in F_a, and from the relation $\pi^T \left(x^D, y^D\right) - \pi^T \left(x^{C(t)}, y^{C(t)}\right) = 0$ we get

$$\pi_s + \sum_{w \in C(t)} \pi_{sw} = 0. \tag{9}$$

When t is a leaf node of the tree equation (9) reduces to $\pi_s + \pi_{st} = 0$ such that $\pi_{st} = 0$. Choosing t by successively moving inwards in G_T from the leaf nodes we obtain that $\pi_{st} = 0$ for all $t \in V_T \setminus \{i, u, v\}$.

(e) Next we shall show that $\pi_{is} = \sigma$, where node s is a member of R_u, $u \in L_i$, or R_{uv}, $uv \in T_i$, as above. If $s \in R_u$ let $v \in I_i$ be any inner neighbor of node i. Consider the partitions $C(i), C(u)$ and $C'(i), C'(v)$ of V_T that result from

deleting the edges iu, iv from E_T, respectively. Let $D(i) = V_T \setminus (C(u) \cup C'(v))$ and let $D(i,s) = D(i) \cup \{s\}$. It is easy to verify that the incidence vectors of the cliques induced by $D(i,s)$, as well as $C(u)$ and $C'(i)$, are contained in F_a. From the relation $\pi^T \left(x^{D(i,s)}, y^{D(i,s)}\right) - \pi^T \left(x^{C'(i)}, y^{C'(i)}\right) = 0$ we get

$$\pi_s + \sum_{w \in D(i)} \pi_{sw} - \pi^T \left(x^{C(u)}, y^{C(u)}\right) - \sum_{e \in \delta(C(u), D(i))} \pi_e = 0.$$

By (c), (d), and (b) this reduces to $\pi_{si} - \pi_{iu} = 0$. So $\pi_{is} = \pi_{iu} = \sigma$. This proves that $\pi_{is} = \sigma$ for all $s \in R_{uv}$, $uv \in T_i$, and all $s \in R_u$, $u \in L_i$.

(f) Let i, s, u, v be distinct nodes defined as above. We shall show that $\pi_{su} = -\sigma$ for all $s \in R_{uv}$, $uv \in T_i$, and $s \in R_u$, $u \in L_i$. If $s \in R_u$ let $v \in I_i$ be an inner neighbor of node i. Deleting the edge iv from E_T we obtain the node set $C(i)$ which contains nodes i and u. Let $D = C(i) \cup \{s\}$. Then the incidence vectors of the cliques induced by $C(i)$ and D are members of F_a, and it follows that

$$\pi_s + \sum_{w \in C(i)} \pi_{sw} = 0.$$

This reduces to $\pi_{si} + \pi_{su} = 0$, and from (e) we get $\pi_{su} = -\pi_{is} = -\sigma$.

(g) Let $s, t \in R_{uv}$, $uv \in T_i$, or $s, t \in R_u$, $u \in L_i$, be two distinct nodes. If $s, t \in R_u$, let $v \in I_i$. Consider the node set $D(i,s)$ which is obtained as in (e) above by adding node s to the set $D(i) = V_T \setminus (C(u) \cup C'(v))$. Now let $D(i,s,t) = D(i,s) \cup \{t\}$. Then F_a contains both incidence vectors of the cliques induced by $D(i,s)$ and $D(i,s,t)$, and it follows that

$$\pi_t + \sum_{w \in D(i,s)} \pi_{tw} = 0.$$

This reduces to $\pi_{it} + \pi_{st} = 0$, and then we obtain $\pi_{st} = -\pi_{it} = -\sigma$. This proves that $\pi_e = -\sigma$ for all $e \in E_n(R_{uv}) \cup E_n(R_u)$.

(h) Denote by Z the union of the nodes in all the sets R_{uv}, R_ℓ. It is trivial to establish that $\pi_{st} = 0$ for all node pairs s and t that belong to different node sets, e.g. $s \in R_e$ and $t \in R_f$ for $e \neq f$.

This establishes that $\pi_v = \sigma a_v$ for all $v \in V_T \cup Z$ and $\pi_e = \sigma a_e$ for all $e \in E_n(V_T \cup Z)$. Furthermore, it is easy to verify that $\pi_v = 0$ for all $v \in V_n \setminus (V_T \cup Z)$ and $\pi_e = 0$ for all $e \in E_n \setminus E_n(V_T \cup Z)$. This completes the proof. □

2.2 Multistar inequalities

A multistar in K_n consists of two disjoint sets of nodes U, V and edge set $\delta(U, V) \cup E_n(V)$. The node set V constitutes the nucleus of the multistar and the nodes in U are called satellites. We have found some new classes of facet defining inequalities for $P_n(b)$ whose support graphs are multistars. These inequality classes have been discovered by means of the same technique, and therefore we open this section with an outline of this approach.

The structure of a particular class of the multistar inequalities is obtained by adding together clique- or cut inequalities that are partially overlapping. More specifically, let U' and V be disjoint sets of nodes and, for each $u \in U'$, consider the clique inequality, respectively a cut inequality, on nodes $V \cup \{u\}$. Adding together these inequalities we get a valid inequality whose support is the multistar with nucleus V and satellites U'. This multistar inequality is clearly not facet defining for $P_n(b)$, but in some cases — when $|U'|$ is sufficiently large with respect to b — it provides a basic structure that can be utilized with an augmented set of satellites $U \supset U'$. This simple modification frequently yields a facet defining multistar inequality.

We have found one class of facet defining multistar inequalities by applying this approach to clique inequalities and two further inequality classes by different applications of the approach to cut inequalities. These inequality classes are the subjects of the following sections.

2.2.1 Clique-star inequalities

The structure of the *clique-star inequalities* that we consider in this section has been derived from b partially overlapping clique inequalities (using $\beta = 1$). These inequalities are not completely new, since Hunting et al. [5] have provided the subclass of the inequalities (4) where the nucleus consists of a single node.

Theorem 2 *Let $S, T \subset V_n$ be two disjoint, nonempty subsets of nodes. Then the clique-star inequality*

$$bx(T) + x(S) - by(E_n(T)) - y(\delta(S, T)) \leq b \tag{10}$$

is valid for $P_n(b)$. It defines a facet of $P_n(b)$, for $b \geq 3$, if and only if $|S| \geq b+1$, and $S = V_n \setminus T$ when $|T| = 1$.

Proof. We denote the inequality by $a^T(x,y) \leq b$. Let $C \subset V_n$ be any subset of nodes such that $|C| \leq b$, and let $s_C = |S \cap C|$ and $t_C = |T \cap C|$. In order to prove validity we shall show that $a^T\left(x^C, y^C\right) - b \leq 0$. From (10) we get

$$\begin{aligned} a^T\left(x^C, y^C\right) - b &= b\left(t_C - \frac{1}{2}t_C(t_C-1)\right) + s_C - s_C t_C - b \\ &= b\left(t_C - \frac{1}{2}t_C(t_C-1) - 1\right) + s_C(1-t_C). \end{aligned}$$

Suppose that $t_C = 0$. Then $a^T\left(x^C, y^C\right) - b = -b + s_C \leq 0$, because $s_C \leq b$. Alternatively, we have $t_C \geq 1$. Then $s_C(1-t_C) \leq 0$, and

$$\begin{aligned} a^T\left(x^C, y^C\right) - b &\leq b\left(t_C - \frac{1}{2}t_C(t_C-1) - 1\right) \\ &= -\frac{1}{2}b\left(t_C^2 - 3t_C + 2\right) \\ &= -\frac{1}{2}b(t_C-1)(t_C-2) \\ &\leq 0. \end{aligned}$$

This proves that the clique-star inequalities are valid for $P_n(b)$.

Facet: We first prove the necessary conditions. We require $|S| > b$ because otherwise the inequality can be obtained as the sum of other valid inequalities. Suppose that $T = \{v\}$ and $S \subset V_n \setminus \{v\}$. In this latter case the inequality can be obtained as the sum of the inequality (4) and the inequalities $y_{uv} - x_u \leq 0$ for all $u \in V_n \setminus (S \cup T)$. This proves necessity.

In order to prove sufficiency assume that the conditions of the theorem are satisfied. Let $\pi^T(x,y) \leq \pi_0$ be a facet defining inequality for $P_n(b)$ such that $\pi^T(x,y) = \pi_0$ for all $(x,y) \in P_n(b)$ that satisfy (10) with equality. We shall show that $(\pi, \pi_0) = (\sigma a, \sigma b)$ for some $\sigma \in \mathbb{R}_+$. We do this by exhibiting node sets C of feasible cliques such that $a^T\left(x^C, y^C\right) = b$. Then we use the relations $\pi^T\left(x^C, y^C\right) = \pi_0$ to derive the desired result.

(a) From $C = \{t\}$, $t \in T$, we get $\pi_t = \pi_0$ for all $t \in T$.

(b) From $C = \{t, u\}$, $t, u \in T$, we then get $\pi_{tu} = -\pi_0$ for all $tu \in E_n(T)$.

(c) From $C = \{s, t\}$, $s \in S$, $t \in T$, it follows that $\pi_s = -\pi_{st}$. Furthermore, this relation is independent of the particular node $t \in T$. So for every $s \in S$ we get $\pi_s = -\pi_{st} = \sigma_s$ for all $t \in T$.

(d) Then, from $C = \{t, u, v\}$, $u, v \in S$, $t \in T$, it follows that $\pi_{uv} = 0$ for all $uv \in E_n(S)$.

(e) Let $S' \subset S$ such that $|S'| = b - 1$, and let $u, v \in S \setminus S'$ be two distinct nodes. Let $C = \{u\} \cup S'$ and $D = \{v\} \cup S'$. Then from $\pi^T \left(x^C, y^C\right) - \pi^T \left(x^D, y^D\right) = 0$ we get

$$\pi_u - \pi_v + \sum_{s \in S'} (\pi_{su} - \pi_{sv}) = \pi_u - \pi_v = 0$$

such that $\pi_u = \pi_v = \sigma_v$. It follows that $\pi_s = \sigma$ for all $s \in S$ and $\pi_e = -\sigma$ for all $e \in \delta(S, T)$.

(f) From $C = \{u\} \cup S'$ as in (e) it then follows that $\pi_0 = b\sigma$.

This proves that (10) defines a facet of $P_n(b)$ when $S \cup T = V_n$. So now we consider zero lifting for the cases where $\bar{V} = V_n \setminus (S \cup T) \neq \emptyset$. Note that $|T| \geq 2$ in these cases by assumption.

(g) From $C = \{t, v\}$, $t \in T$, $v \in \bar{V}$, we get $\pi_v + \pi_{tv} = 0$. Similarly, $\pi_v + \pi_{uv} = 0$ for $u \in T$. Now, from $D = \{t, u, v\}$ we get $\pi_v + \pi_{tv} + \pi_{uv} = 0$. It follows that $\pi_v = \pi_{tv} = 0$ for all $t \in T$ and all $v \in \bar{V}$.

(h) From $C = \{t, u, v\}$, $t \in T$, $u, v \in \bar{V}$, it then follows that $\pi_{uv} = 0$ for all $uv \in E_n(\bar{V})$.

(i) From $C = \{s, t, v\}$, $s \in S$, $t \in T$, $v \in \bar{V}$, it follows that $\pi_{sv} = 0$ for all $s \in S$ and all $v \in \bar{V}$.

We have now shown that $\pi = \sigma a$ and $\pi_0 = \sigma b$. This completes the proof that the clique-star inequality defines a facet of $P_n(b)$. □

2.2.2 Cut-star inequalities

The *cut-star inequalities* considered in this section are derived from $b-1$ partially overlapping cut inequalities. Let R, S, T be three mutually disjoint node sets, and for each $r \in R$ consider the cut inequality induced by node sets S and $T \cup \{r\}$. Combining these inequalities we obtain an inequality whose support is a multistar with nucleus $S \cup T$ and satellites R.

Theorem 3 *Let $R, S, T \subset V_n$ be three mutually disjoint, nonempty subsets of nodes. Then the cut-star inequality*

$$\begin{aligned}(b-1)[\, y(\delta(S,T)) - x(S) - y(E_n(S)) - y(E_n(T))\,] \\ + y(\delta(R,S)) - y(\delta(R,T) \leq 0\end{aligned} \tag{11}$$

is valid for $P_n(b)$. It defines a facet of $P_n(b)$, for $b \geq 3$, if and only if i) $|R| \geq b$ and ii) $b \geq 4$ and $|T| \geq 2$ when $|S| \geq 2$.

Proof. For simplicity we denote the inequality by $a^T(x,y) \leq 0$. Let $C \subset V_n$ with $|C| \leq b$ be the node set of any feasible clique in K_n. In order to prove that the cut-star inequality is valid for $P_n(b)$ we shall show that $a^T\left(x^C, y^C\right) \leq 0$. Let $r_C = |R \cap C|$, $s_C = |S \cap C|$, and $t_C = |T \cap C|$. From (11) we get

$$\begin{aligned} a^T\left(x^C, y^C\right) &= (b-1)\left[s_C t_C - s_C - \frac{1}{2}s_C(s_C-1) - \frac{1}{2}t_C(t_C-1)\right] \\ &\quad + r_C s_C - r_C t_C \\ &= (b-1)\left[s_C t_C - \frac{1}{2}s_C(s_C+1) - \frac{1}{2}t_C(t_C-1)\right] \\ &\quad + r_C(s_C - t_C). \end{aligned}$$

It immediately follows that $a^T\left(x^C, y^C\right) \leq 0$ when $s_C = 0$. So we may assume that $s_C \geq 1$, which in turn implies $r_C \leq b-1$.

Suppose that $s_C - t_C \geq 0$. Then $r_C(s_C - t_C) \leq (b-1)(s_C - t_C)$, and

$$\begin{aligned} a^T\left(x^C, y^C\right) &\leq (b-1)\left[s_C t_C - \frac{1}{2}s_C(s_C+1) - \frac{1}{2}t_C(t_C-1) + s_C - t_C\right] \\ &= -\frac{1}{2}(b-1)\left[t_C^2 - (2s_C-1)t_C + s_C(s_C-1)\right] \\ &= -\frac{1}{2}(b-1)(t_C - s_C + 1)(t_C - s_C) \\ &\leq 0. \end{aligned}$$

On the other hand, suppose that $s_C - t_C \leq 0$. This implies that $r_C(s_C - t_C) \leq 0$, and

$$\begin{aligned} a^T\left(x^C, y^C\right) &\leq (b-1)\left[s_C t_C - \frac{1}{2}s_C(s_C+1) - \frac{1}{2}t_C(t_C-1)\right] \\ &= -\frac{1}{2}(b-1)\left[t_C^2 - (2s_C+1)t_C + s_C(s_C+1)\right] \\ &= -\frac{1}{2}(b-1)(t_C - s_C)(t_C - s_C - 1) \\ &\leq 0. \end{aligned}$$

Hence, the cut-star inequality is valid for $P_n(b)$.

Facet: The condition $|R| \geq b$ is necessary, because otherwise the cut-star inequality can be obtained as the sum of other valid inequalities. When $|S| \geq 2$ and $b \leq 3$ or $|T| = 1$ we have $y_e = 0$ for all $e \in E_n(S)$ in every incidence vector of a feasible clique. This implies that the face of $P_n(b)$ defined by the cut-star

inequality is properly contained in the face that is defined by the nonnegativity constraint $y_e \geq 0$, $e \in E_n(S)$. See also the proof of [11, Theorem 3.2].

We now assume that the conditions of the theorem are satisfied. As in the previous proof we shall show that there exists $\sigma \in \mathbb{R}_+$ such that $(\pi, \pi_0) = (\sigma a, 0)$, where $\pi^T(x,y) \leq \pi_0$ is a facet defining inequality for $P_n(b)$ with the property that $a^T(x,y) = 0 \Rightarrow \pi^T(x,y) = \pi_0$.

(a) Since $a^T\left(x^C, y^C\right) = 0$ for $C = \emptyset$, we immediately get $\pi_0 = 0$.

(b) From $C = \{v\}$, $v \in R \cup T$, it follows that $\pi_v = 0$ for all $v \in R \cup T$.

(c) From $C = \{u, v\}$, $u, v \in R$, we then get $\pi_{uv} = 0$ for all $uv \in E_n(R)$.

(d) Let $R' \subset R$ such that $|R'| = b - 2$, let $u, v \in R \setminus R'$ be distinct nodes, and let $s \in S$. Set $C = \{s, u\} \cup R'$ and $D = \{s, v\} \cup R'$. From the relation $\pi^T\left(x^C, y^C\right) - \pi^T\left(x^D, y^D\right) = 0$ we get

$$\pi_u - \pi_v + \pi_{su} - \pi_{sv} + \sum_{r \in R'} (\pi_{ru} - \pi_{rv}) = \pi_{su} - \pi_{sv} = 0,$$

which shows that $\pi_{su} = \pi_{sv}$. This implies that for every $s \in S$ we have $\pi_{sv} = \sigma_s$ for all $v \in R$.

(e) From $C = \{s, u\} \cup R'$ as in (d) it then follows that $\pi_s + (b-1)\sigma_s = 0$. Hence, $\pi_s = -(b-1)\sigma_s$ for all $s \in S$.

(f) Now, from $C = \{s, t\}$, $s \in S$, $t \in T$, we get $\pi_s + \pi_{st} = 0$. So, for every $s \in S$, $\pi_{st} = (b-1)\sigma_s$ for all $t \in T$.

(g) From $C = \{s, u, v\}$, $s \in S$, $u, v \in T$, it follows that $\pi_{uv} = -(b-1)\sigma_s$ for all $uv \in E_n(T)$.

(h) The relation in (g) is clearly independent of any particular node $s \in S$. This implies that $\sigma_s = \sigma$ for all $s \in S$.

(i) From $C = \{r, s, t\}$, $r \in R$, $s \in S$, $t \in T$, we get $\pi_{rt} = -\pi_{rs} = -\sigma$ for all $rt \in \delta(R, T)$.

(j) Suppose that $|S| \geq 2$ and let $C = \{i, j, u, v\}$, $i, j \in S$, $u, v \in T$. Then it follows that $(b-1)\sigma + \pi_{ij} = 0$. Hence, $\pi_e = -(b-1)\sigma$ for all $e \in E_n(S)$.

We have now proven that the cut-star inequality defines a facet of $P_n(b)$ when $R \cup S \cup T = V_n$. In order to obtain the result when $\bar{V} = V_n \setminus (R \cup S \cup T) \neq \emptyset$ it must be shown that $\pi_v = 0$ for all $v \in \bar{V}$ and $\pi_e = 0$ for all $e \in E_n(\bar{V}) \cup \delta(\bar{V}, R \cup S \cup T)$. Since this is very easy to establish, we leave out this part of the proof. □

We note that (5) is the special case of the cut-star inequality where $S = \{s\}$, $T = \{t\}$, and $R = V_n \setminus \{s, t\}$.

2.2.3 T-star inequalities

We have derived one further inequality class from partially overlapping cut inequalities. The structure of the following *T-star inequalities* has been obtained by adding $b-2$ cut inequalities that overlap in node set T and may also overlap in node set S. Let R, S and T be mutually disjoint node sets where S may be empty. For each $r \in R$ consider the cut inequality induced by node sets $\{r\} \cup S$ and T. Combining these inequalities we get an inequality whose support graph is a multistar with nucleus $S \cup T$ and satellites R.

Theorem 4 *Let $R, S, T \subset V_n$ be three mutually disjoint subsets of nodes with $|T| \geq 2$. Then the T-star inequality*

$$\begin{aligned}(b-2)[\, y(\delta(S,T)) - x(S) - y(E_n(S)) - y(E_n(T))\,] \\ - x(R) - y(\delta(R,S)) + y(\delta(R,T)) \leq 0\end{aligned} \tag{12}$$

is valid for $P_n(b)$. It defines a facet of $P_n(b)$, for $b \geq 3$, if and only if $|R| \geq b-1$, and $b \geq 4$ when $|S| \geq 1$.

Proof. Denote the inequality by $a^T(x,y) \leq 0$. We first show that the inequality is valid. Let $C \subset V_n$ be any feasible subset of nodes, and let $r_C = |R \cap C|$, $s_C = |S \cap C|$ and $t_C = |T \cap C|$. From the structure of (12) we have

$$\begin{aligned}a^T\left(x^C, y^C\right) &= (b-2)\left[s_C t_C - s_C - \frac{1}{2}s_C(s_C-1) - \frac{1}{2}t_C(t_C-1)\right] \\ &\quad - r_C - r_C s_C + r_C t_C \\ &= (b-2)\left[s_C t_C - \frac{1}{2}s_C(s_C+1) - \frac{1}{2}t_C(t_C-1)\right] \\ &\quad + r_C(t_C - s_C - 1).\end{aligned}$$

Since $a^T\left(x^C, y^C\right) \leq 0$ when $t_C = 0$, we assume that $t_C \geq 1$. First suppose that $s_C \geq 1$. When $t_C - s_C - 1 \geq 0$ we have $r_C(t_C - s_C - 1) \leq (b-2)(t_C - s_C - 1)$ such that

$$\begin{aligned}a^T\left(x^C, y^C\right) &\leq (b-2)\left[s_C t_C - \frac{1}{2}s_C(s_C+1) - \frac{1}{2}t_C(t_C-1) + t_C - s_C - 1\right] \\ &= -\frac{1}{2}(b-2)\left[t_C^2 - (2s_C+3)t_C + s_C(s_C+3) + 2\right] \\ &= -\frac{1}{2}(b-2)(t_C - s_C - 1)(t_C - s_C - 2) \\ &\leq 0.\end{aligned}$$

On the other hand, when $t_C - s_C - 1 \leq 0$ we have $r_C(t_C - s_C - 1) \leq 0$ such that

$$\begin{aligned} a^T\left(x^C, y^C\right) &\leq (b-2)\left[s_C t_C - \frac{1}{2}s_C(s_C+1) - \frac{1}{2}t_C(t_C-1)\right] \\ &= -\frac{1}{2}(b-2)\left[t_C^2 - (2s_C+1)t_C + s_C(s_C+1)\right] \\ &= -\frac{1}{2}(b-2)(t_C - s_C)(t_C - s_C - 1) \\ &\leq 0. \end{aligned}$$

Finally, suppose that $s_C = 0$. Then $r_C(t_C - 1) \leq (b - t_C)(t_C - 1)$ because $r_C \leq b - t_C$. Therefore,

$$\begin{aligned} a^T\left(x^C, y^C\right) &\leq -\frac{1}{2}(b-2)t_C(t_C-1) + (b-t_C)(t_C-1) \\ &= -\frac{1}{2}b(t_C-2)(t_C-1) \\ &\leq 0. \end{aligned}$$

This proves validity.

Facet: The T-star inequality can be obtained as the sum of other valid inequalities when $|R| \leq b-2$. When $b = 3$ the T-star inequality with coefficients $a_e = 0$ for all $e \in \delta(R, S)$ is valid for $P_n(b)$; it is the T-star inequality obtained by setting $R := R \cup S$ and $S := \emptyset$. This proves necessity.

In order to prove sufficiency we assume that the conditions of the theorem are satisfied. Let $\pi^T(x, y) \leq \pi_0$ be a facet defining inequality such that $a^T(x, y) = 0 \Rightarrow \pi^T(x, y) = \pi_0$. As usual we shall prove that $\pi = \sigma a$ for some $\sigma \in \mathbb{R}_+$.

(a) Since $a^T\left(x^C, y^C\right) = 0$ for $C = \emptyset$, we get $\pi_0 = 0$.

(b) From $C = \{t\}$, $t \in T$, we get $\pi_t = 0$ for all $t \in T$.

(c) Let $C = \{r, t\}$, $r \in R$, $t \in T$. Then it follows that, for every $r \in R$, we have $\pi_{rt} = -\pi_r$ for all $t \in T$.

(d) From $C = \{i, j, t\}$, $i, j \in R$, $t \in T$, it follows that $\pi_{ij} = 0$ for all $ij \in E_n(R)$.

(e) Let $R' \subset R$ such that $|R'| = b - 3$, and let $i, j \in R \setminus R'$ and $t, u \in T$ be distinct nodes. Let $C = \{i, t, u\} \cup R'$ and $D = \{j, t, u\} \cup R'$. From the relation $\pi^T\left(x^C, y^C\right) - \pi^T\left(x^D, y^D\right) = 0$ we get

$$\pi_i - \pi_j + \pi_{it} + \pi_{iu} - \pi_{jt} - \pi_{ju} + \sum_{r \in R'}(\pi_{ir} - \pi_{jr}) = 0,$$

which reduces to $-\pi_i + \pi_j = 0$ by (c) and (d). So $-\pi_j = -\pi_i = \pi_{it} := \sigma$. Hence, $\pi_e = \sigma$ for all $e \in \delta(R,T)$ and $\pi_r = -\sigma$ for all $r \in R$.

(f) Let $C = \{i,t,u\} \cup R'$ as in (e). Then it follows that $(b-2)(2\sigma-\sigma)+\pi_{tu} = 0$, which implies that $\pi_e = -(b-2)\sigma$ for all $e \in E_n(T)$.

Now we assume that S in nonempty.

(g) From $C = \{s,t\}$, $s \in S$, $t \in T$, we get $\pi_{st} = -\pi_s$ for all $t \in T$.

(h) From $C = \{s,t,u\}$, $s \in S$, $t,u \in T$, it then follows that $\pi_s = \pi_{tu} = -(b-2)\sigma$. Hence, $\pi_s = -(b-2)\sigma$ for all $s \in S$ and $\pi_e = (b-2)\sigma$ for all $e \in \delta(S,T)$.

(i) From $C = \{i,j,t,u\}$, $i,j \in S$, $t,u \in T$, it follows that $\pi_{ij} + (b-2)\sigma = 0$. This implies that $\pi_e = -(b-2)\sigma$ for all $e \in E_n(S)$.

(j) From $C = \{r,s,t,u\}$, $r \in R$, $s \in S$, $t,u \in T$, it follows that $\pi_{rs} + \sigma = 0$. So $\pi_e = -\sigma$ for all $e \in \delta(R,S)$.

We have established so far that $\pi_v = \sigma a_v$ for all $v \in U = R \cup S \cup T$ and $\pi_e = \sigma a_e$ for all $e \in E_n(U)$. It still remains to be shown that $\pi_v = 0$ for all $v \in V_n \setminus U$ and $\pi_e = 0$ for all $e \in E_n \setminus E_n(U)$. However, these latter relations are easily established. So we immediately jump to conclude that $(\pi, \pi_0) = (\sigma a, 0)$. This completes the proof. □

3 A branch-and-cut algorithm

We utilize the above classes of inequalities in a branch-and-cut algorithm for the WCP. A branch-and-cut algorithm is a branch-and-bound algorithm that incorporates a fractional cutting-plane algorithm for the solution of the subproblem LPs of the branch-and-bound enumeration. We also use a heuristic in order to obtain an initial feasible clique. In this section we describe the most important components of our branch-and-cut algorithm.

3.1 A heuristic for the WCP

The purpose of the heuristic is to provide a lower bound on the value of an optimal clique in K_n. This value is used to prune branches of the branch-and-cut enumeration tree such that the optimization process at a branch is terminated as soon as the LP objective value falls below this bound. The heuristic is a two phase heuristic composed of a construction heuristic and an improvement heuristic.

The construction heuristic starts with a single node and successively builds a clique by adding further nodes, one node at a time. This is done in a greedy manner by including the next node that gives the largest contribution to the current clique, where the contribution is the weight of the node plus the sum of the weights of the edges between this node and the nodes in the clique. We have designed this heuristic to take into account that some or all of the node weights may be negative. For this reason a node may be included in the clique although the partial contribution from this node is negative; it is hoped that the contribution becomes positive subsequently after the inclusion of further nodes. So further nodes are included in the clique until it consists of b nodes. Finally, the clique values for the first p nodes included are compared for $p = 1, \ldots, b$, and only the first k of these b nodes that give the highest clique value are kept in the clique. In this way the heuristic may provide cliques with fewer than b nodes.

The improvement heuristic is an exchange heuristic like the one described in Späth [14]. Given a clique on node set C it calculates, for all node pairs $u \in C$ and $v \in V_n \setminus C$, the gain that results from replacing u by v in C. An exchange of the node pair that gives the largest gain is made if the gain is positive. This is repeated until no positive gain can be obtained by exchanging two nodes.

We use each node of the graph as a starting node for the construction heuristic, and for each clique found in this manner we also apply the improvement heuristic. So we find n cliques (some of which may turn out to be identical) and use the one with the largest value as a candidate optimal clique.

3.2 Fractional cutting plane algorithm

The fractional cutting plane algorithm (FCPA) calls a collection of separation procedures in order to identify facet defining inequalities that cut off the current fractional LP solution, adds such inequalities to the LP, reoptimizes the revised LP, and removes from the LP inequalities that have become nonbinding. This section deals with the management of the inequalities that are generated by the separation procedures and the control of constraint generation. In section 3.3 we give detailed descriptions of some of the separation procedures.

The FCPA uses three lists to keep track of the generated inequalities according to their status of being either violated, active, or removed:

- A list of candidate inequalities that are violated by the current LP solution.

- A list of active inequalities of the current LP.
- A constraint pool containing inequalities that have been removed from the LP, but are kept to be used again.

A particular inequality is only contained in one of these lists, and it is represented by a class identifier and an array consisting of one or more ordered sets of node indices.

The separation procedures are very often able to find a lot of violated inequalities. So in order to keep the number of candidate inequalities at a manageable level the constraint identification is terminated whenever $40n$ inequalities have been found. All newly generated inequalities are placed in the list of candidate inequalities. This list consists of 10 "buckets" each of which contains inequalities with similar degrees of violation (this approach is taken from Grötschel and Wakabayashi [3]).

The addition of violated inequalities to the LP and LP reoptimization are performed iteratively. A maximum number of 400 inequalities are added to the LP at a time. So up to 400 inequalities are taken from the most violated buckets of the candidate list and are appended to the list of active inequalities. After the augmented LP has been reoptimized any still violated inequalities in the candidate list are sorted anew into the appropriate buckets; the rest of the inequalities are discarded from further consideration. As long as any inequalities in the candidate list remain violated the addition of inequalities and LP reoptimization are repeated. We shall refer to this iterative process as the reoptimization loop.

In order to keep the LPs small — and make reoptimizations fast — nonbinding inequalities are removed from the LP. This is done every time the LP has been reoptimized, unless only a few inequalities have been added since the previous deletion of constraints. The nonbinding inequalities are removed from the list of active inequalities, and some of them are placed in the constraint pool as explained below. We consider an inequality nonbinding when the associated slack variable is in the LP basis, irrespective of its value.

The constraint pool is used to store inequalities for which a heuristic separation procedure is used. The main purpose of the constraint pool is to have a place to save currently inactive inequalities that may be needed to re-establish the upper bound at a node of the branch-and-cut enumeration tree. Whenever branching takes place we make a list of pointers to the currently binding inequalities that are not separated by an exact procedure. This list is saved at the branching node such that any necessary inequalities can easily be retrieved from

the pool, or they can be generated when needed. However, to the extent that there is idle space, the pool is also used for temporary storage of inequalities that have been active recently. In this way we get easy access to some inequalities that cannot be guaranteed to be identified otherwise.

Now we describe the control of constraint identification. The very first LP that is constructed only includes some of the constraints of the formulation (1) of the WCP, namely the $|E_n|$ inequalities $x_u + x_v - y_{uv} \leq 1$ and the n star inequalities together with the nonnegativity bounds $y_e \geq 0$. After the solution of this LP has been obtained further inequalities that are violated by the LP solution are generated and added to the LP.

The generation of different classes or subclasses of violated inequalities is controlled in a hierarchical manner. This hierarchy consists of three levels:

At *Level 1* we identify any violated inequalities that belong to the formulation (1) of the WCP and any violated inequalities (4). At this level constraint identification and LP reoptimizations are repeated until all inequalities are satisfied by the current LP solution. Then we go to Level 2 if any node variables x_v take on fractional values; otherwise the FCPA terminates with a feasible integer LP solution.

At *Level 2* we identify any violated triangle cut inequalities $y_{uv} + y_{uw} - y_{vw} - x_u \leq 0$ and triangle clique inequalities $x_u + x_v + x_w - y_{uv} - y_{uw} - y_{vw} \leq 1$. Whenever branching has taken place we also check the violation of the inequalities in the pool that were binding at the parent node of the enumeration tree. If any violated inequalities are identified at this level we complete the reoptimization loop and then immediately return to Level 1; otherwise we go to Level 3.

At *Level 3* we attempt to identify violated α-inequalities, clique inequalities, various multistar inequalities, and cut-tree inequalities. We also make a full scan of the constraint pool. In case any violated inequalities are found the reoptimization loop is completed and then we return to Level 1. Otherwise the FCPA terminates.

We have implemented one further termination mechanism for the FCPA. This is when tailing off occurs, i.e. when several successive LP objective values only decrease little. In order to determine when tailing off occurs we use a prespecified tolerance ϵ and an iteration limit ℓ. Tailing off detection begins at Level 2 of the FCPA, where LP objective values at the end of successive reoptimization loops are compared. If the relative difference between two successive LP objective values does not exceed the tolerance ϵ a tailing off indicator is turned on. This indicator is turned off subsequently whenever this difference

exceeds ϵ. If the indicator is still on after ℓ successive reoptimization loops the optimization process at Level 2 of the FCPA is terminated. At Level 3 the tailing off detection continues in a similar manner. After the reoptimization loop at Level 3 has been completed the tailing off indicator is turned off if the reduction rate of the LP objective value exceeds ϵ. In any case the FCPA returns to levels 1 and 2, but now Level 2 terminates sooner. If the tailing off indicator is still on after the first reoptimization loop at Level 2 control is passed back to Level 3. This process is repeated as long as the tailing off indicator remains turned on, except that after ℓ successive Level 3 entries the FCPA gives up and terminates due to tailing off.

We have experienced that the choice of the numerical values that are assigned to the parameters ϵ and ℓ are critical for the overall performance of the branch-and-cut algorithm. In fact, our computational experiments show that highly different values of ϵ should be used for different sets of test problems. We shall explain the particular settings of the parameters we have used when we present the computational results.

3.3 Separation procedures

This section describes the separation procedures that are used in the FCPA to identify violated inequalities. Because some of the procedures are rather trivial, a detailed description of these separation procedures will not be given here. We just note that the inequalities in (1) and (4) that are used at Level 1 of the FCPA can be separated exactly by complete enumeration with complexity $O(n^2)$. Similarly, the triangle cut- and clique inequalities that are used at Level 2 can be separated exactly with complexity $O(n^3)$.

The input to the separation procedures is the current LP solution $(\bar{x}, \bar{y})$. The output is a number of inequalities that are violated by this solution and which are placed in the list of candidate inequalities. Since some of the separation procedures may find identical inequalities, any particular violated inequality is only saved in the candidate list if it is not already contained in the appropriate bucket. In this way we avoid the use of several identical inequalities.

3.3.1 Separation of α-inequalities

Besides the exact procedure for generating triangle cut inequalities that is used at Level 2 in the FCPA we have implemented a heuristic separation procedure

for the identification of violated α-inequalities (2) with $|T| \geq 3$ or $|S| \geq 2$. This separation procedure is called by the FCPA at Level 3.

We utilize an easy method to calculate the integer value of the α coefficient that gives maximal violation of the inequality for any given node sets S and T. The best α is obtained by setting

$$\alpha = \lceil \bar{x}(T) - \bar{x}(S) \rceil . \tag{13}$$

This result can be derived by a marginal analysis of the violation. The violation of the α-inequality is

$$\text{viol}(\alpha) = \alpha \left(\bar{x}(T) - \bar{x}(S) \right) - \alpha(\alpha - 1)/2 + k,$$

where $k = \bar{y}(\delta(S,T)) - \bar{y}(E_n(S)) - \bar{y}(E_n(T)) - \bar{x}(T)$. As noted by Macambira and de Souza [8] viol(α) is a second-degree polynomial function in α. Now we note that the violation of the inequality can be increased by incrementing α by 1 if viol$(\alpha + 1) >$ viol(α). This is equivalent to $\bar{x}(T) - \bar{x}(S) > \alpha$. Alternatively, the violation can be increased by decrementing α by 1 if viol$(\alpha - 1) >$ viol(α). This is equivalent to $\bar{x}(T) - \bar{x}(S) + 1 < \alpha$. It follows that the coefficient α that gives maximal violation satisfies $\bar{x}(T) - \bar{x}(S) \leq \alpha \leq \bar{x}(T) - \bar{x}(S) + 1$ for given S and T.

The separation procedure works on the currently active α-inequalities, and the idea is to make minor modifications of these inequalities that result in new violated inequalities. Since active inequalities are satisfied with equality, it is relatively easy to identify such attractive modifications. We consider any active α-inequality given by the triple $(S, T, \bar{\alpha})$.

The first modification involves calculating the currently best α using (13). If $\alpha \neq \bar{\alpha}$ we obtain a new violated inequality by using this coefficient. It should be noted, however, that $\alpha \leq 0$ implies that the roles of node sets S and T should be interchanged. In this latter case we simply generate a new violated inequality given by $(T, S, 1)$, but only when $|S| \geq 2$.

The second modification that is attempted is to augment one of the node sets S and T by a single node that is not already contained in these sets. When an attractive augmentation is considered the best α is recalculated such that $\bar{\alpha}$ may be incremented when T is augmented or decremented when S is augmented.

It is possible in many cases to identify several new violated inequalities from a single active inequality by these modifications. However, we have chosen to generate only one new inequality for each active α-inequality. So as soon as a violated inequality has been found we stop considering any other modifications of the current inequality.

3.3.2 Separation of clique inequalities

As mentioned above we use an exact procedure to identify any violated triangle clique inequalities at Level 2 of the FCPA. At Level 3 we use a heuristic separation procedure that modifies currently active clique inequalities in a manner that is very similar to the modification of α-inequalities.

For any given node set T there is a closed-form formula for the value of β that gives maximal violation. This is $\beta = \lfloor \bar{x}(T) \rfloor$, which follows directly from (13) with $\beta = \alpha - 1$ and $\bar{x}(S) = 0$. So the first modification that is considered is to calculate the currently best β using this formula. This is done for all active clique inequalities with $|T| \geq 4$. If the best β differs from the one used we obtain a new violated inequality by using this coefficient.

Whenever the first modification is not successful we attempt to augment node set T by one node. We consider every node $v \in V_n \setminus T$ for which $\bar{x}_v > 0$ and use $\beta = \lfloor \bar{x}(T) + \bar{x}_v \rfloor$. Every violated clique inequality that is found in this way is saved in the list of candidate inequalities (if not already present). So we may generate several new inequalities from one active clique inequality.

3.3.3 Separation of cut-tree inequalities

We have implemented two separation procedures for the class of cut-tree inequalities that are used at Level 3 of the FCPA. One procedure is a greedy construction heuristic, while the other one attempts to modify currently active cut-tree inequalities. The procedures are very similar in that they only differ in the identification of a tree $G_T = (V_T, E_T)$. The modification procedure simply uses the tree in the support of a currently active inequality, whereas the construction heuristic builds a new tree.

The construction heuristic builds a tree in the following way. We start with an initial tree that is an edge of the graph, $V_T = \{u, v\}$ and $E_T = \{uv\}$. Subsequently this tree is augmented iteratively by adding a node $w \in V_n \setminus V_T$ and an edge vw such that $v \in V_T$ and $\bar{y}_{vw} - \bar{x}_v$ is maximal. This continues until V_T has $b + 1$ nodes. We use each edge $uv \in E_n$ with $\bar{y}_{uv} > 0$ as an initial tree such that several distinct trees may be generated in this manner. Any trees that turn out to be stars are discarded from further consideration.

Once a tree is available we determine what additional node sets should be associated with the tree and the nodes to be included in these sets. Both the modification procedure and the construction heuristic do this in the same way.

For every inner node $i \in I$ of the tree we have an (initially empty) node set

R_ℓ for each leaf neighbor $\ell \in L_i$ and a node set R_{uv} for each edge uv of the neighbor tree T_i that spans the inner neighbors I_i of i. However, when $|I_i| \geq 3$ we have a choice as to which edges should be included in T_i. We determine T_i as a maximal spanning tree in $E_n(I_i)$ based on weights for the edges $uv \in E_n(I_i)$ that are calculated as follows. Any single node $w \in V_n \setminus V_T$ that is included in R_{uv} contributes with $\gamma(w) := \bar{y}_{iw} - \bar{y}_{uw} - \bar{y}_{vw}$ to the left-hand side value of the cut-tree inequality. So we simply use the sum of all positive $\gamma(w)$ as weights for the edges uv, although this sum actually overestimates the potential contribution from node set R_{uv} to the violation of the inequality.

After the neighbor trees of the inner nodes have been determined we assign nodes to the sets R_ℓ, R_{uv} which are empty initially. This is done in a greedy manner, one node at a time. First we identify the best node set for each node $w \in V_n \setminus V_T$; this is the set for which $\gamma(w)$ is maximal (in the case of a leaf neighbor set R_ℓ we calculate $\gamma(w) := \bar{y}_{iw} - \bar{y}_{\ell w}$). In this way we tentatively assign every node to a particular node set, provided that the corresponding γ contribution is positive. If $\gamma(w)$ is nonpositive for all node sets, node w will not be assigned to any set. Subsequently some of the nodes are permanently assigned to their node set. Among all tentatively assigned nodes we select the one that has maximal γ contribution. Suppose that this is node s. Then s is permanently assigned to its node set R, meaning that the set is augmented, $R := R \cup \{s\}$. This assignment changes the contributions from the remaining nodes that are tentatively assigned to set R. So we update $\gamma(w) := \gamma(w) - \bar{y}_{sw}$ for all these nodes, and if $\gamma(w)$ becomes nonpositive we delete w from the list of tentatively assigned nodes. The permanent assignments are continued in this manner as long as there are any tentatively assigned nodes left.

Upon completion of the above node set augmentations the left-hand side value of the corresponding cut-tree inequality is tested. If it is positive we have a violated inequality which is placed in the list of candidate inequalities, provided that we do not find an inequality with the same description in the appropriate bucket of the list.

3.3.4 Separation of multistar inequalities

The separation procedures for the three classes of multistar inequalities are quite similar. So here we shall only describe the separation of cut-star inequalities (11); the separation of clique-star inequalities and T-star inequalities works analogously. However, we remark that in the present algorithm we only utilize T-star inequalities (12) with an empty node set S. The multistar inequalities are used

at Level 3 in the FCPA.

First we note that for a given nucleus a most violated cut-star inequality can be identified with complexity $O(n)$ simply by using all satellites that give positive contributions to the left-hand side value of the inequality (in fact, this is true for all multistar inequalities). We utilize this property in all separation procedures that are described here.

We make exact separation of cut-star inequalities with $|S| = |T| = 1$. This is a very simple procedure where we consider all edges $uv \in E_n$ for which $\bar{y}_{uv} > 0$ and identify the best set of satellites R for each of the two choices $S = \{u\}$, $T = \{v\}$ and $S = \{v\}$, $T = \{u\}$, respectively.

We have also implemented two heuristic separation procedures that work on the currently active inequalities. These procedures identify cut-star inequalities with $|S| \geq 2$ or $|T| \geq 2$. In one procedure we use the currently binding α-inequalities. The nodes in the support of the α-inequality are used as the nucleus for the cut-star inequality with nodes sets S and T keeping their respective roles. In the other separation procedure we try to modify currently active cut-star inequalities. The first modification consists in identifying a new best set of satellites. The second modification is to augment node set T by the satellite node with largest contribution, while the last modification that is attempted is to augment node set S by one node (under the condition that $|T| \geq 2$).

3.4 Branching

Branching is performed whenever the FCPA terminates with a noninteger LP solution whose objective value exceeds the value of the candidate optimal clique. Branching is based on variable dichotomy. We select a node variable x_v with fractional value and create two branches: the down-branch where we create a subproblem with the additional restriction that $x_v = 0$, and the up-branch where we require $x_v = 1$.

In the presence of several fractional valued node variables we choose one which is expected to harm the LP objective value most. We identify the most fractional valued node variables, i.e. the variables whose values are closest to 0.5. For all these variables we calculate the sum of the weight of the corresponding node and the weights of all edges incident to the node. We choose the node variable for which this sum is largest. In a more pure form this method of branching variable selection is referred to as enhanced branching in Linderoth and Savelsbergh [7].

We use the best-bound search strategy in the selection of the next branch to process. Furthermore, whenever $b > n/2$ we first solve the subproblem at the up-branch; otherwise we first solve the problem at the down-branch. This rule is based on the likelihood of finding an optimal integer solution at the branch in question; in the former case we expect to find an optimal solution at an up-branch more often than not. Identifying an optimal solution at an early stage of the branch-and-cut enumeration saves work in the FCPA, because it can be terminated as soon as the LP objective value reaches the lower bound.

4 Computational results

In this section we summarize the computational results we have obtained by applying the branch-and-cut algorithm to two sets of test problems. The first set of test problems has been provided by Macambira and de Souza [8] and has also been used in Hunting et al. [5]. We provide a second set of test problems. These are instances of a column generation problem in the context of graph partitioning.

The algorithm has been implemented in a C program using CPLEX 5.0 as the LP solver. All computational results have been obtained by running our program under Windows 95 on a 350 MHz PC.

4.1 Macambira's and de Souza's test problems

This set of test problems consists of 60 problem instances. The problems are associated with (complete) graphs on $n \in \{40, 42, 44, 45, 46, 48\}$ nodes, and the maximal clique size is fixed at $b = \lfloor n/2 \rfloor$. The edge weights are randomly generated; half of the problem instances have positive weights in the interval $[1, 1000]$, and the other half of the instances have positive and negative weights in the interval $[-1000, 1000]$. The sizes of the edge weights are controled by a parameter $k \in \{1, \ldots, 5\}$. All node weights are zero.

Table 1 shows the computational results for the problem instances with positive edge weights, and table 2 shows the results for the instances with negative and positive weights. The tables are to be read as follows. The first three columns state the values of n, b, and k. The next three columns state the value of the clique found by the heuristic, the final LP objective value obtained by the FCPA at the root (first) node of the enumeration tree, and the optimal clique value. The seventh column gives the number of branches considered during the

n	b	k	Heuristic	First node	Optimal	Branches	Time (s)
40	20	1	109346	109346.0	109346	1	103
40	20	2	82451	82451.0	82451	1	46
40	20	3	68759	68759.0	68759	1	131
40	20	4	60782	60782.0	60782	1	114
40	20	5	60513	60513.0	60513	1	51
42	21	1	120299	120299.0	120299	1	201
42	21	2	87810	87810.0	87810	1	215
42	21	3	76554	76554.0	76554	1	163
42	21	4	69482	69482.0	69482	1	102
42	21	5	67383	67383.0	67383	1	52
44	22	1	136525	136525.0	136525	1	133
44	22	2	98186	98186.0	98186	1	284
44	22	3	84675	84675.0	84675	1	169
44	22	4	75274	75274.0	75274	1	214
44	22	5	69540	69540.0	69549	1	110
45	22	1	138694	138694.0	138694	1	245
45	22	2	98321	98321.0	98321	1	369
45	22	3	82644	82743.0	82743	1	312
45	22	4	77500	77500.0	77500	1	225
45	22	5	69563	69563.0	69563	1	291
46	23	1	142985	142985.0	142985	1	415
46	23	2	108243	108243.0	108243	1	439
46	23	3	94859	94859.0	94859	1	232
46	23	4	78747	78747.0	78747	1	419
46	23	5	72290	72431.0	72431	1	370
48	24	1	163397	163397.0	163397	1	386
48	24	2	115471	115475.4	115471	3	1135
48	24	3	96567	96666.0	96666	1	517
48	24	4	88728	88728.0	88728	1	272
48	24	5	82117	82117.0	82117	1	262

Table 1: Macambira and de Souza problems: positive weights.

branch-and-cut enumeration. A 1 in this column means that branching was not necessary. The last column states the total computation time in seconds.

The tailing off parameters (cf. page 23) used to obtain these results were set to a tolerance of $\epsilon = 0.001$ and $\ell = 4$ iterations. These settings of the parameters were chosen in order to avoid branching for the large majority of the problem instances. Extensive experimentation clearly suggests that this is the most efficient approach for these problem instances.

A direct comparision with the computation times quoted in Hunting et al. [5] and Macambira and de Souza [8] is not possible because different computer systems have been used. Nevertheless, we feel confident about claiming that our branch-and-cut algorithm performs better than the algorithms presented in these papers. Hunting et al. have used a computer with a 125 MHz processor. Using the MHz-figures as a measure of relative speed we estimate that our computer is approximately 3 times faster. So we have divided the computation times in [5] by 3 and compared them with the above results. For the problem instances in table 1 our algorithm is about 5 times faster on the average, ranging from 2 to 12 times; and for the problem instances in table 2 we get a factor of 10 in our favor, ranging from 3 to 26.

Although Macambira and de Souza have also used a branch-and-cut approach, the computation times they obtain do not appear to be better than those of Hunting et al. We think that the relatively poor performance of their algorithm has two explanations. One explanation is that they only utilize cut- and α-inequalities in their FCPA. This is indeed a very strong and useful class of inequalities; together with the constraints in (1) they are sufficient to solve the above problem instances without branching. However, we have found that the subclass (4) of the clique-star inequalities is also extremely useful. Utilizing these few inequalities, as we do at Level 1, considerably speeds up the convergence of our FCPA. The other explanation is that they do not delete unbinding inequalities from the LPs. It is apparent from the computational results in [8] that the LPs grow very large in terms of the number constraints. This has an adverse effect on the reoptimization speed of the LP solver. In our FCPA we solve many, many more LPs, but they are also much smaller, and reoptimizations are faster.

4.2 Column generation problems for graph partitioning

In this section we present the computatinal results for some instances of a column generation problem for graph partitioning. The framework for solving graph

n	b	k	Heuristic	First node	Optimal	Branches	Time (s)
40	20	1	70348	70348.0	70348	1	223
40	20	2	45404	45404.0	45404	1	68
40	20	3	34091	34091.0	34091	1	64
40	20	4	27758	27758.0	27758	1	84
40	20	5	27967	27967.0	27967	1	44
42	21	1	81633	81633.0	81633	1	269
42	21	2	46828	46828.0	46828	1	193
42	21	3	36689	36689.0	36689	1	84
42	21	4	35987	35987.0	35987	1	58
42	21	5	35460	35460.0	35460	1	59
44	22	1	90620	90620.0	90620	1	347
44	22	2	56960	56960.0	56960	1	195
44	22	3	40697	40697.0	40697	1	151
44	22	4	32601	32601.0	32601	1	169
44	22	5	29407	29407.0	29407	1	129
45	22	1	102295	102295.0	102295	1	252
45	22	2	55103	55103.0	55103	1	353
45	22	3	43914	43914.0	43914	1	84
45	22	4	33543	33990.0	33990	1	140
45	22	5	30974	30974.0	30974	1	237
46	23	1	99550	99550.0	99550	1	383
46	23	2	58361	58361.0	58361	1	358
46	23	3	43915	43915.0	43915	1	242
46	23	4	32968	32968.0	32968	1	344
46	23	5	31000	31000.0	31000	1	144
48	24	1	113478	113478.0	113478	1	800
48	24	2	61768	61768.0	61768	1	840
48	24	3	45941	45941.0	45941	1	290
48	24	4	36903	36903.0	36903	1	206
48	24	5	31351	31351.0	31351	1	307

Table 2: Macambira and de Souza problems: negative and positive weights.

partitioning problems via column generation is described in Johnson et al. [6]. Mehrotra [9] studies a special case of the column generation problem that he formulates as a cardinality constrained Boolean quadratic problem (CBQP). This problem is the same as the WCP, except that edge variables are only defined for edges with nonzero weights.

The instances of the CBQP that are considered in [9] are interesting from a computational point of view, because Mehrotra states that *"Our experience ... indicates that these problems are very difficult to solve."* Unfortunately, the data from that study is no longer available. Instead we have chosen to generate some new problem instances that are based on the same graphs which are from [6]. For this reason the problem instances we obtain should be quite similar to the original problem instances. However, we treat these problems as instances of the WCP by assigning zero weights to all edges that are missing in the graphs.

The problem we consider is to generate columns for the LP relaxation of a set packing problem (SPP). This SPP has n rows, and each column corresponds to a clique such that the coefficient in the ith row is 1 if and only if the ith node is in the clique; otherwise it is 0. The objective coefficient of the column is the sum of the weights of all edges in the clique. We must generate a column with maximal reduced cost. This problem is a WCP with edge weights as described and a weight for every node which is the negative of the dual variable for the corresponding row of the SPP.

We have obtained our instances of this problem in the following way. For each graph and each maximal clique size $b \in \{10, 15, 20, 25\}$ we used our heuristic for the WCP to find an intial clique and then constructed the LP relaxation of the SPP with the correspondig column. At subsequent iterations the SPP was reoptimized, the new dual solution was used to provide new node weights, and the heuristic was applied again (actually, this heuristic was extended by also applying the exchange heuristic to the existing columns of the SPP). As long as the heuristic was capable of finding a clique with positive value the corresponding column was added to the SPP, and this process continued. At the stages in the process, where the heuristic failed to find a positive valued clique, we have saved the dual solution and thus obtained a new instance of the WCP. This new problem instance was solved by the branch-and-cut algorithm. In case a positive valued clique was found the corresponding column was added to the SPP, and the whole process was repeated by reapplying the heuristic. Only when the optimal clique value was zero the whole process stopped.

The computational results for the problem instances that were solved by

branch-and-cut are shown in table 3. The column 'Graph' refers to the graph number used in [6]. 'Instance' gives the order in which a problem instance was obtained for any given graph and b. The other columns are as in tables 1 and 2, except that the heuristic clique values are omitted; they are all 0 by virtue of the above process.

In order to get the results in table 3 we had to change the settings of the tailing off parameters. We have used a much larger tolerance $\epsilon = 0.05$ and the same iteration limit $\ell = 4$. This means that the FCPA terminates due to tailing off much earlier than otherwise, that the LP objective values provided by the FCPA are larger, and that branching becomes more extensive. However, for the majority of these problem instances this is advantageous. The larger computational efforts of the FCPA that result from a small value of ϵ do not pay off in terms of smaller computation times due to less extensive branching. This is very likely because some other classes of facet defining inequalities are needed for these problem instances, or because the separation procedures currently employed are too simple to identify the right inequalities.

On the other hand, we would like to emphasize that the above problem instances probably are some of the most difficult ones in this context. It has been pointed out in [6] and [9] that as the reduced costs approach zero the column generation problems become harder, because the gap between the the LP value (at the root node) and the optimal value tends to get larger.

In accordance with this observation the problem instances that have large optimal values (reduced costs) are relatively easy to solve. We have also solved the column generation problems for the above graphs with zero node weights using our branch-and-cut algorithm. This corresponds to the problem of finding a best first column to add to the SPP. All these latter problem instances were easily solved without branching. The longest computation time was 9 seconds for the Graph 6 instance with $b = 25$.

As a final remark we think it is worth noting that these graphs are sparse in the sense that only about ten percent of the edges have nonzero weights. Despite this fact the utilization of the complete graph formulation (1) of the problem seems very appropriate. The overhead that is incurred by the definition and use of additional edge variables with zero weights appears to be compensated for by the tightness of the formulation.

Graph	n	b	Instance	First node	Optimal	Branches	Time (s)
2	30	10	1	1.0	1.0	1	4
2	30	10	2	0.0	0.0	1	6
2	30	15	1	5.6	0.0	3	69
2	30	20	1	6.2	0.0	9	113
2	30	25	1	5.4	0.0	9	151
1	45	10	1	11.5	0.0	7	235
1	45	15	1	8.4	0.0	3	196
1	45	20	1	33.9	0.0	3	156
1	45	25	1	14.0	9.1	5	428
1	45	25	2	13.7	0.0	7	578
3	47	10	1	3.7	0.0	7	537
3	47	15	1	7.3	5.3	3	316
3	47	15	2	8.0	0.0	9	683
3	47	20	1	31.7	0.0	11	900
3	47	25	1	38.8	0.0	11	1773
4	47	10	1	4.6	0.0	7	294
4	47	15	1	6.7	0.0	7	617
4	47	20	1	11.1	0.0	7	809
4	47	25	1	20.3	0.0	13	1084
6	61	10	1	37.5	0.0	9	908
6	61	15	1	94.8	0.0	15	1829
6	61	20	1	70.4	0.0	11	2772
6	61	25	1	243.5	0.0	15	3553

Table 3: Column generation problems for Johnson et al. graphs.

Acknowledgements

I kindly thank Professor Cid de Souza for providing his and Dr. Macambira's test problems for me.

References

[1] G. Dijkhuizen, U. Faigle, A cutting-plane approach to the edge-weighted maximal clique problem, European Journal of Operational Research 69 (1993) 121–130.

[2] U. Faigle, R. Garbe, K. Heerink, B. Spieker, LP-relaxations for the edge-weighted subclique problem, in: A. Bachem, U. Derigs, M. Jünger, R. Schrader (Eds.), Operations Research '93, Physica-Verlag, Heidelberg, 1994, pp. 157–160.

[3] M. Grötschel, Y. Wakabayashi, A cutting plane algorithm for a clustering problem, Mathematical Programming 45 (1989) 59–96.

[4] M. Hunting, Relaxation Techniques for Discrete Optimization Problems — Theory and Algorithms, Ph.D. thesis, University of Twente, 1998.

[5] M. Hunting, U. Faigle, W. Kern, A Lagrangian relaxation approach to the edge-weighted clique problem, European Journal of Operational Research 131 (2001) 119–131.

[6] E.L. Johnson, A. Mehrotra, G.L. Nemhauser, Min-cut clustering, Mathematical Programming 62 (1993) 133–151.

[7] J.T. Linderoth, M.W.P. Savelsbergh, A computational study of search strategies for mixed integer programming, INFORMS Journal on Computing 11 (1999) 173–187.

[8] E.M. Macambira, C.C. de Souza, The edge-weighted clique problem: Valid inequalities, facets and polyhedral computations, European Journal of Operational Research 123 (2000) 346–371.

[9] A. Mehrotra, Cardinality constrained Boolean quadratic polytope, Discrete Applied Mathematics 79 (1997) 137–154.

[10] M. Padberg, The Boolean quadric polytope: Some characteristics, facets and relatives, Mathematical Programming 45 (1989) 139–172.

[11] K. Park, K. Lee, S. Park, An extended formulation approach to the edge-weighted maximal clique problem, European Journal of Operational Research 95 (1996) 671–682.

[12] H.D. Sherali, Y. Lee, W.P. Adams, A simultaneous lifting strategy for identifying new classes of facets for the Boolean quadric polytope, Operations Research Letters 17 (1995) 19–26.

[13] M.M. Sørensen, b-tree facets for the simple graph partitioning polytope, Working paper 00-4, Dept. of Management Science and Logistics, The Aarhus School of Business, 2000.

[14] H. Späth, Heuristically determining cliques of given cardinality and with minimal cost within weighted complete graphs, Zeitschrift für Operations Research 29 (1985) 125–131.