COMBINATORICA 6 (3) (1986) 221—233

PACKING AND COVERING A TREE BY SUBTREES

I. BÁRÁNY, J. EDMONDS and L. A. WOLSEY

Received 27 September 1984

For two polyhedra associated with packing subtrees of a tree, the structure of the vertices is described, and efficient algorithms are given for optimisation over the polyhedra. For the related problem of covering a tree by subtrees, a reduction to a packing problem, and an efficient algorithm are presented when the family of trees is "fork-free".

1. Introduction

Given a tree R with vertex set V and a family of its subtrees $\mathcal{F}$, consider the problem of packing these subtrees into the tree:

$$(P_0) \qquad \text{Max} \sum \{c(T)x(T): T \in \mathcal{F}\}$$
$$\text{subject to} \quad \sum \{x(T): T \ni v\} \leqq 1 \qquad (v \in V),$$
$$x(T) \in \{0, 1\} \quad (T \in \mathcal{F}),$$

where $c(T)$ is a weight function on $\mathcal{F}$.

It is known that this problem can be solved by a polynomial-time algorithm as the intersection graph of $\mathcal{F}$ is triangulated [3] and the maximum weighted stable set problem on a triangulated graph can be solved efficiently [2], see also [4].

In this paper we consider various generalisations of problem P_0. For related problems involving all subtrees of a tree, see [5].

In order to state our results we need some preparation. First we assume that the tree is rooted (with root $r \in V$). This induces a partial ordering of V in the usual way: we say that $u < v$ $(u, v \in V)$ if $u \neq v$ and u belongs to the unique path connecting r and v. We are also given a monotone non-decreasing real valued function $a: V \to \mathbf{R}$, i.e., $u, v \in V$, $u < v$ implies $a(u) \leqq a(v)$.

The first problem we consider is the linear program

$$(P_1) \qquad \text{Max} \sum \{c(T)x(T): T \in \mathcal{F}\}: x \in Q_1$$

where $Q_1 \subset R_+^{|\mathcal{F}|}$ is described by the inequalities:

$$\sum \{x(T): T \ni v\} \leqq a(v) \quad (v \in V),$$
$$x(T) \geqq 0 \quad (T \in \mathcal{F}).$$

AMS subject classification (1980): 68 E 10, 05 C 05, 05 B 40

We describe a dual greedy algorithm for P_1, which can also be viewed as a dynamic programming algorithm. This is one way to obtain a characterisation of the vertices of Q_1, which are integer whenever $a: V \to \mathbf{R}$ is an integer valued monotone nondecreasing function.

The second problem we consider is also a packing problem, except that the family $\mathcal{F}$ of subtrees is replaced by the family of all subtrees, and a somewhat special objective function. We consider the linear program:

$$(P_2) \qquad \text{Max} \sum_{u,v \in V} c(u,v)x(u,v): \; x \in Q_2$$

where $Q_2 \subseteqq \mathbf{R}^{|V|^2}$ is described by the inequalities

$$\sum_{u \in V} x(u,v) \leqq a(v) \quad (v \in V),$$

$$x(u,v) - x(u,w) \leqq 0 \quad \text{if} \quad w \in [u,v] \quad (u,v \in V)$$

$$x(u,v) \geqq 0 \quad (u,v \in V),$$

where there is a variable $x(u,v)$ for each pair $u, v \in V$, and where $[u,v]$ denotes the vertices on the path connecting u and v.

Take a solution $x(u,v)$ to P_2. Fix u and set $\alpha_0 = \min\{x(u,v): v \in V, x(u,v) > 0\}$ and $S_0^u = \{v \in V: x(u,v) > 0\}$. If $S_0^u \neq \emptyset$, then S_0^u is a tree containing u. Now define inductively $S_i^u = \{v \in V: x(u,v) > \alpha_{i-1}\}$ and $\alpha_i = \min\{x(u,v): x(u,v) > \alpha_{i-1}\}$ unless $S_i^u = \emptyset$. Clearly S_i^u is a tree containing u and $x(u,v) = \alpha_0 s_0^u(v) + (\alpha_1 - \alpha_0) s_1^u(v) + (\alpha_2 - \alpha_1) s_2^u(v) + \ldots$ where s_i^u is the characteristic function of S_i^u, i.e., $s_i^u(v) = 1$ if $v \in S_i^u$ and 0 otherwise. So we may think of a solution $x(u,v)$ of P_2 as a weighted sum of subtrees S_i^u of R, each S_i^u being "rooted" at u. Relative to problem (P_1), here $\mathcal{F}$ is the set of all subtrees but the objective value associated with a subtree T is

$$\max_{u \in T} \{\sum c(u,v): \; v \in T\}.$$

Here again we obtain a characterisation of the vertices of Q_2, and a dual greedy algorithm to solve the linear program. The problem P_2 with $a(v) = 1$ for all $v \in V$ was the major motivation for this work, as it generalises the tree packing problems considered in [1].

The third problem we consider is that of covering the tree R by subtrees from $\mathcal{F}$. This can be formulated as the integer program:

$$(C_0) \qquad \min \sum \{c(T)x(T): \; T \in \mathcal{F}\}$$

$$\sum \{x(T): \; T \ni v\} \geqq 1 \quad (v \in V)$$

$$x(T) \in \{0,1\} \quad (T \in \mathcal{F}).$$

While C_0 is generally NP-hard, we show that when the family $\mathcal{F}$ of subtrees has a certain property, denoted *forkfree*, which generalises a property of distance subtrees, problem C_0 can be reduced to problem P_2 and hence solved efficiently.

We mentioned already that P_0 is a maximum weighted stable set problem on a triangulated graph. It is shown in Section 5 that the two problems are equivalent.

It is known that when the constraint matrix appearing in C_0 is totally balanced, C_0 can be solved by linear programming. What is the relationship between

totally balanced matrices and node versus subtree incidence matrices when the subtree family is forkfree? This and similar questions are discussed in Section 5. In addition we return to P_2 via the problem that was the starting point of this research, the economic lot sizing problem with backlogging.

2. Packing with a given family of subtrees

Some notation is needed. A subtree and its set of vertices are denoted by the same letter. For any subtree T of R we call $r(T)=\text{Min}\{v: v\in T\}$ the root of T. This is clearly well-defined. For $v\in V$ we denote $R(v)$ the subtree spanned by the vertices $\{u\in V: u\geqq v\}$. Note that if $u<v$, then $[u, v]=\{w\in V: u\leqq w\leqq v\}$. The predecessor $p(v)$ of node v is the first vertex on the path going from v to r. The successor set $S(v)$ of vertex v is the set of vertices $w\in V$ having v as a predecessor. The successor set $S(T)$ of a tree T is the set of vertices $w\notin T$ with $p(w)\in T$.

We assume that $a(v)\geqq 0$ for each $v\in V$, as otherwise Q_1 and Q_2 are empty. Finally, for $v\in V$ we write $\mathcal{F}(v)=\{T\in\mathcal{F}: r(T)=v\}$.

Here we consider the problem P_1, and the associated polytope Q_1. We shall also need the linear programming dual of P_1:

$$(D_1)\qquad \min \sum_{v\in V} a(v)y(v)$$

$$\sum\{y(v): v\in T\}\geqq c(T)\quad (T\in\mathcal{F})$$

$$y(v)\geqq 0\quad (v\in V).$$

First we describe an algorithm for D_1.

The dual greedy algorithm for D_1

All vertices are initially unmarked. Set $d(T)\leftarrow c(T)$ for $T\in\mathcal{F}$.

i) Choose a vertex $v\in V$ that is unmarked but all $w\in R(v)\setminus\{v\}$ are marked.

ii) Fix dual variables

Set $y(v)\leftarrow \max\{0, \max\{d(T): T\in\mathcal{F}(v)\}\}$

$$= \begin{cases} 0 \text{ if } \mathcal{F}(v)=\emptyset \text{ or } d(T)\leqq 0 \text{ for all } T\in\mathcal{F}(v) \\ d(T_v) \text{ for some } T_v\in\mathcal{F}(v) \text{ otherwise.}\end{cases}$$

Mark v. For later use we fix $T_v\in\mathcal{F}(v)$ with $y(v)=d(T_v)$ if $y(v)>0$, ties are broken arbitrarily.

iii) Update $d(T)$, $T\in\mathcal{F}$.

Set $d(T)\leftarrow d(T)$ if $T\in\mathcal{F}$, $v\notin T$.

Set $d(T)\leftarrow d(T)-y(v)$ if $T\in\mathcal{F}$, $v\in T$.

iv) If r is unmarked, return to i).

Otherwise, stop.

Observe that the vector y constructed this way does not depend on the dual objective function, $a(v)$, $v\in V$. Note also that on termination of the algorithm y is dual feasible and $d(T)\leqq 0$ for each $T\in\mathcal{F}$. These facts follow by an easy induction argument.

The greedy algorithm for P_1

Here we make use of the subtrees T_v that were fixed during the dual greedy algorithm.

All vertices are initially unmarked. Set $b(v) \leftarrow a(v)$ for $v \in V$.

i) Choose a vertex $v \in V$ that is
   a) unmarked
   b) for which there is no unmarked vertex $w \prec v$.

ii) Fix primal variables.
   If $y(v)=0$, set $x(T)=0$ for all $T \in \mathcal{F}(v)$.
   If $y(v)>0$, set $x(T_v)=b(v)$, $x(T)=0$ for $T \in \mathcal{F}(v) \setminus T_v$.
   Mark vertex v.

iii) Update $b(u)$, $u \in V$.
   If $y(v)=0$, b is unchanged.
   If $y(v)>0$, set $b(u) \leftarrow b(u)-x(T_v)$, $u \in T_v$.
   $b(u) \leftarrow b(u)$, otherwise.

iv) If all nodes are marked, stop.
   If not, return to i).

Theorem 1. *If* $y \in \mathbf{R}_+^{|V|}$ *and* $x \in \mathbf{R}_+^{|\mathcal{F}|}$ *are constructed by the dual greedy and the greedy algorithm respectively,* y *is optimal in* D_1 *and* x *is optimal in* P_1.

Proof. It is easily seen that y and x are dual and primal feasible as $d(T) \geqq 0$, $T \in \mathcal{F}$, and $b(u) \geqq 0$, $u \in V$ respectively. From step iii) of the greedy algorithm for P_1, we see that $y(v)>0$ only if $b(v)=0$. Also $x(T)>0$ only if $d(T)=0$. Hence complementary slackness holds and both solutions are optimal. ∎

An alternative view of the algorithm used in solving P_1 and D_1 will be useful later. Let $G(v)$ be the optimal value of P_1 when $a(u)=0$ for all $u \in V \setminus R(v)$, $a(u)=1$ for all $u \in R(v)$. The associated solutions are called 1-packings of $R(v)$. In other words $G(v)$ is the optimal value of a 1-packing of R when restricted to the trees $T \in \mathcal{F}$ that lie in $R(v)$.

A recursion for P_1

$$G(v) = \operatorname{Max}\Big\{ \sum_{w \in S(v)} G(w),\ \operatorname*{Max}_{T \in \mathcal{F}(v)} \big(c(T) + \sum_{w \in S(T)} G(w)\big) \Big\}.$$

This says that there are essentially two possibilities. Either the optimal solution contains no tree $T \in \mathcal{F}$ including the vertex v. In this case the solution must consist of optimal 1-packings of each of the trees $R(w)$, $w \in S(v)$. Alternatively if some tree T containing v is used in the optimal solution, the remainder of the solution must consist of optimal 1-packings of the trees $R(w)$, $w \in S(T)$.

The connection between the recursive algorithm and the dual greedy algorithm is readily seen by observing, that

$$y(v) = G(v) - \sum_{w \in S(v)} G(w) \tag{1}$$

satisfies

$$y(v) = \operatorname{Max}\{0, \operatorname{Max}_{T \in \mathcal{F}(v)} (c(T) - \sum_{w \in T \setminus \{v\}} y(w))\}$$

which coincides with the dual variables given by the dual greedy algorithm.

Now let x^v denote an optimal 1-packing of $R(v)$ with value $G(v)$.

Claim. *Let*

$$x = \sum_{v \in V} x^v (a(v) - a(p(v)))$$

where $a(p(r)) = 0$ *by definition, and* y *be defined by* (1), *then* $x \in \mathbf{R}_+^{|\mathcal{F}|}$ *and* $y \in \mathbf{R}_+^{|V|}$ *are optimal solutions to* P_1 *and* D_1 *respectively.*

Proof. Clearly $x \geqq 0$ as $x^v(T) \geqq 0$ and $a(v) \geqq a(p(v))$ for $v \in V$. To see that x satisfies $\sum\{x(T): T \ni w\} \leqq a(w)$, note that $\sum\{x^v(T): T \ni w\} \leqq 1$ and $\sum\{x^v(T): T \ni w\} = 0$ if $v \not\geqq w$ and hence

$$\sum \Big\{ \sum_{v \in V} x^v(T)(a(v) - a(p(v))): T \ni w \Big\} = \sum_{v \geqq w} \sum_{T \ni w} x^v(T)(a(v) - a(p(v))) \leqq$$

$$\leqq \sum_{v \geqq w} [a(v) - a(p(v))] \leqq a(w).$$

Its value is $\sum_{v \in V} G(v)(a(v) - a(p(v)))$.

To show its optimality note that $y(v)$ is a dual feasible solution of value

$$\sum_{v \in V} a(v) y(v) = \sum_{v \in V} a(v)(G(v) - \sum_{w \in S(v)} G(w)) = \sum_{v \in V} G(v)(a(v) - a(p(v))). \blacksquare$$

We have now shown

Theorem 2. *Every vertex of* Q_1 *is of the form*

$$x = \sum_{v \in V} x^v (a(v) - a(p(v))). \blacksquare$$

3. Packing with all possible subtrees

Here we consider problem P_2. The approach is similar to that of the previous section. We use a dual greedy or recursive algorithm to calculate the optimal value $H(v)$ of a packing for P_2 when $a(u) = 1$ for $u \in R(v)$ and $a(u) = 0$ otherwise, and let $x^v \in \mathbf{R}^{|V|^2}$ be the associated optimal 1-packing. The family of 1-packings x^v, $v \in V$ is then used to construct an optimal packing for P_2 for any $a: V \to R^1$ that is nondecreasing outwards from r.

A recursive algorithm for P_2

In order to calculate $H(v)$ recursively starting from the leaves and working inward towards the root, we use an auxiliary function $\varphi_u(v)$ which is essentially the optimal value of a 1-packing of $R(v)$ with the extra condition $x(u, v) = 1$. But these is another point of view in looking at $\varphi_u(v)$ that will be useful. Remember that the tree R is rooted at r but, as we mentioned in the introduction, it is conve-

nient to think of a solution $x(u, v)$ of P_2 as a weighted sum of subtrees S_i^u "rooted" at node u. If the solution $x(u, v)$ takes 0-1 values only (which will be the case with the 1-packing of $R(v)$), then where is at most one subtree S^u rooted at u and $S^u \cap \cap S^v = \emptyset$ for $u \neq v$. In this case we say that the subtree S^u covers the node v provided $x(u, v) = 1$. More precisely, we define $\varphi_u(v)$ as follows: If $u \geqq v$, $\varphi_u(v)$ is the optimal weight of a 1-packing of $R(v)$ where v is covered by a subtree S^u rooted at u. If $u \not\geqq v$, then $\varphi_u(v)$ is the optimal weight of a 1-packing of $R(v) \cup [u, v]$ where v is covered by a subtree S^u rooted at u, and only the values of covering vertices of $R(v)$ are counted. The formal definition is this:

$$\text{if } u \geqq v,\ \varphi_u(v) = \max \sum_{w \in R(v)} \sum_{z \in R(v)} c(w, z)\, x(w, z) : x \in Q_2,$$
$$x(u, v) = 1,\ x(w, z) = 0 \text{ if } w \notin R(v) \text{ or if } z \notin R(v)$$
$$x(w, z) \in \{0, 1\} \quad \text{otherwise}$$
$$\text{if } u \not\geqq v,\ \varphi_u(v) = \max \sum_{w \in R(v) \cup \{u\}} \sum_{z \in R(v)} c(w, z)\, x(w, z) : x \in Q_2,$$
$$x(u, v) = 1,\ x(w, z) = 0 \text{ if } w \notin R(v) \cup \{u\} \text{ or } z \notin R(v)$$
$$x(w, z) \in \{0, 1\} \quad \text{otherwise.}$$

Now we obtain:

$$H(v) = \operatorname{Max}\Big\{ \sum_{w \in S(v)} H(w), \operatorname*{Max}_{u \in R(v)} \varphi_u(v) \Big\}$$

because in an optimal 1-packing of $R(v)$ either no subtree covers vertex v, or vertex v is covered by a subtree rooted at u where $u \in R(v)$.

The recursion for $\varphi_u(v)$ is given by:

$$\varphi_u(v) = c(u, v) + \sum_{w \in S(v)} \operatorname{Max}\{\varphi_u(w), H(w)\} \quad \text{if} \quad u \notin R(v) \setminus \{v\},$$

$$\varphi_u(v) = c(u, v) + \sum_{w \in S(v) \setminus \{w'\}} \operatorname{Max}\{\varphi_u(w), H(w)\} + \varphi_u(w') \quad \text{if} \quad u \in R(v) \setminus \{v\}$$

where $w' = S(v) \cap [u, v]$ is the first vertex after v on the path $[v, u]$.

To justify this recursion, we decompose the value of the 1-packing of $R(v)$ into $c(u, v)$ plus the value of the packing of the trees $R(w)$, $w \in S(v)$. When $u \notin \notin R(v) \setminus \{v\}$, the value of the packing of $R(w)$ is $\varphi_u(w)$ if $x(u, w) = 1$, and $H(w)$ if $x(u, w) = 0$, $w \in S(v)$. When $u \in R(v) \setminus \{v\}$, the argument is the same except that $x(u, v) = 1$ implies $x(u, w) = 1$ for all $w \in [u, v]$, and hence the value of the packing of $R(w')$ is $\varphi_u(w')$.

To compute these values we work through the vertices $v \in V$ moving in from the leaves to the root, and for v fixed we first calculate $\varphi_u(v)$ for all $u \in V$ and then $H(v)$. After computing $H(v)$ we compute x^v, the associated optimal 1-packing of $R(v)$ having value $H(v)$. To calculate x^v, we set $x^v(u_0, v) = 1$ if $H(v) = \operatorname*{Max}_{u \in R(v)} \varphi_u(v) = \varphi_{u_0}(v)$ and trace back through $\varphi_{u_0}(v)$, and we set $x^v(z, v) = 0$ for all $z \in V$ if $H(v) = \sum_{w \in S(v)} H(w)$ and then trace back through $H(w)$ for all $w \in S(v)$.

Theorem 3. $x = \sum_{v \in V} x^v(a(v) - a(p(v)))$ *is an optimal solution to* P_2.

Proof. It is readily checked that x belongs to P_2 and its value is $\sum_{v \in V} H(v)(a(v) - a(p(v)))$.

Now we exhibit a dual feasible solution with the same value. The dual of P_2 is

$$\text{Min} \sum_{v \in V} a(v) y(v)$$

$$y(v) + z(u, v) - \sum_{w \in Q(u, v)} z(u, w) \geqq c(u, v) \quad (u, v \in V, \quad u \neq v)$$

$$(D_2) \qquad y(u) - \sum_{w \in Q(u)} z(u, w) \geqq c(u, u) \quad (u \in V)$$

$$y(v), \quad z(u, v) \geqq 0$$

where $Q(u, v)$ is the set of vertices w such that v lies on the path from u to w and is a neighbour of w, and $Q(u) = Q(u, u)$ is the set of neighbours of u. In fact, we again construct the dual solution by a greedy algorithm. Set

$$y(v) = H(v) - \sum_{w \in S(v)} H(w).$$

With this definition P_2 and D_2 have the same value. Now for each $u \in V$, ignore the initial ordering of the vertices, and consider a new ordering in which u is the root. Working in from the leaves to the root u, define

$$z(u, v) = \text{Max} \left\{0, c(u, v) - y(v) + \sum_{w \in Q(u, v)} z(u, w)\right\} \quad \text{for} \quad v \in V.$$

Clearly, $y, z \geqq 0$ and, by construction, the first inequalities of D_2 are satisfied. To show that the second set is satisfied, consider the largest subtree $T \subseteq \{u\} \cup \cup \{v \in V : z(u, v) > 0\}$ containing u. Now

$$c(u, u) - y(u) + \sum_{w \in Q(u)} z(u, w) = \sum_{w \in T} \big(c(u, w) - y(w)\big).$$

But

$$H\big(r(T)\big) \geqq \sum_{v \in T} c(u, v) + \sum_{w \in S(T)} H(w)$$

and hence

$$\sum_{v \in T} \big(c(u, v) - y(v)\big) = \sum_{v \in T} c(u, v) - \sum_{w \in T} y(w)$$

$$= \sum_{v \in T} c(u, v) - \big(H(r(T)) - \sum_{w \in S(T)} H(w)\big) \leqq 0. \blacksquare$$

We mention here that essentially the same method works if in P_2 one or more of the inequalities $\sum_{u \in V} x(u, v) \leqq a(v)$ is replaced by the equality $\sum_{u \in V} x(u, v) = a(v)$. If such a change occurs at a vertex $v \in V$, the recursion for H just changes to

$$H(v) = \text{Max}_{u \in R(v)} \varphi_u(v).$$

4. Covering with a given family of subtrees

Now we consider the problem of finding a minimum value cover of R by subtrees.

$$Z_0 = \text{Min}\left\{\sum c(T)x(T):\ T\in\mathcal{F}\right\}$$

$$(C_0)\qquad \sum\{x(T):\ T\ni v\}\geqq 1 \quad (v\in V)$$

$$x(T)\geqq 0 \quad \text{and} \quad \text{integer} \quad (T\in\mathcal{F}).$$

We assume throughout that $c(T)\geqq 0$ as otherwise $Z_0=-\infty$ or C_0 has no feasible solution.

We also consider two related problems:

(C_1): Find a minimum value partition of R using subtrees $S\subseteq T$, $(T\in\mathcal{F})$ where the cost $c(S, T)$ of each subtree $S\subseteq T$ is $c(T)$. In C_1 one tree $T\in\mathcal{F}$ may contain several subtrees S taking part in the partition. Let Z_1 be the optimal value of C_1. Formally

$$Z_1 = \min\left\{\sum\{c(S, T)x(S, T):\ S\subseteq T,\ T\in\mathcal{F}\}\right. \quad \text{subject to} \quad x(S, T)\in\{0, 1\} \text{ and}$$

$$\left.\sum\{x(S, T):\ v\in S,\ S\subseteq T,\ T\in\mathcal{F}\} = 1 \quad \text{for each} \quad v\in V\right\}.$$

(C_2): Find a minimum value partition of R using at most one subtree $S\subseteq T$ from each $T\in\mathcal{F}$ with cost $c(S, T)=c(T)$. Let Z_2 be the optimal value of C_2. Formally

$$Z_2 = \min\left\{\sum\{c(S, T)x(S, T):\ S\subseteq T,\ T\in\mathcal{F}\}\right. \quad \text{subject to} \quad x(S, T)\in\{0, 1\} \quad \text{and}$$

$$\sum\{x(S, T):\ v\in S,\ S\subseteq T,\ T\in\mathcal{F}\} = 1 \quad \text{for each} \quad v\in V, \quad \text{and}$$

$$\left.\sum\{x(S, T):\ S\subseteq T\}\leqq 1 \quad \text{for each} \quad T\in\mathcal{F}\right\}.$$

Observation. $Z_0\leqq Z_1\leqq Z_2$.

We first show that there is a good algorithm for C_1 with the more general subtree cost function $c(S, T)=c(T)+\sum_{v\in S} c_{T,v}$ if $S\subseteq T$. In fact we reduce C_1 to the problem P_2.

Reduction of C_1 *to* P_2

For each pair (v, T) with $T\in\mathcal{F}$ and $v\in T$, add a dummy vertex $u_{v,T}$ to R, and an edge $(v, u_{v,T})$ between the dummy vertex and v. On the resulting tree, define the values as follows:

$$c(u_{v,T}, u_{v,T}) = -c(T) \quad (v\in T, T\in\mathcal{F})$$

$$c(u_{v,T}, w) = -c_{T,w} \quad (v\in T, T\in\mathcal{F}, w\in T)$$

$$c(u, v) = -\infty \quad \text{otherwise.}$$

The packing problem P_2 on the extended graph with equalities ($=1$) on the vertices of R and inequalities ($\leqq 1$) on the dummy nodes solves C_1.

Now consider the solution of C_2. We say that a set of subtrees $\mathcal{F}^* \subseteq \mathcal{F}$ is a *cover* for R if $\cup\{T: T \in \mathcal{F}^*\} = R$. A family $\mathcal{F}$ of subtrees is said to be *partitionable* if for every cover $\mathcal{F}^* \subseteq \mathcal{F}$ one can find a subtree $S(T) \subseteq T$ for each $T \in \mathcal{F}^*$ such that $\bigcup_{T \in \mathcal{F}^*} S(T) = R$, and if $T_1, T_2 \in \mathcal{F}^*$ with $T_1 \neq T_2$ then $S(T_1) \cap S(T_2) = \emptyset$.

Proposition 4. C_0 *is well-solved for partitionable families, and* $Z_0 = Z_1 = Z_2$.

Proof. As $\mathcal{F}$ is partitionable, every feasible solution of C_0 gives rise to a solution of C_2 of the same value and so $Z_0 = Z_1 = Z_2$. Hence to solve C_0 it suffices to solve C_1 and take as a cover $\mathcal{F}^*$ the set of trees $T \in \mathcal{F}$ used in the solution of C_1. ∎

Therefore we are interested in the existence of partitionable families.

Definition. Trees T_1 and T_2 of R have a *fork* if there are vertices $x_1, y_1 \in T_1 \setminus T_2$ and $x_2, y_2 \in T_2 \setminus T_1$ such that $[x_1, y_1]$ and $[x_2, y_2]$ have a point in common. A family $\mathcal{F}$ of subtrees of R is *fork-free* if no pair $T_1, T_2 \in \mathcal{F}$ has a fork.

Figure 1 shows a tree R, and two subtrees with a fork.

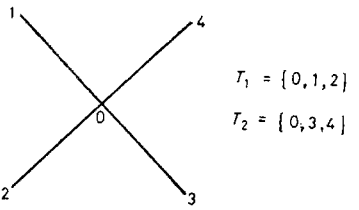

Fig. 1. $T_1 = \{0, 1, 2\}$, $T_2 = \{0, 3, 4\}$

We will prove soon that a fork-tree family is partitionable. The converse is clearly not true. But given a tree R and a family $\mathcal{F}$ of its subtrees and a fixed subtree R' of R, we say that $\mathcal{F}$ is partionable on R' if $\mathcal{F}' = \{T \cap R': T \in \mathcal{F}\}$ is a partitionable family of subtrees of R'. Then one can see easily that $\mathcal{F}$ is fork-tree if it is partitionable on all subtrees R' of R.

Theorem 5. *There exists a "good" algorithm for the covering problem* C_0 *when the family* $\mathcal{F}$ *of subtrees is fork-free.*

Proof. It suffices to show that every fork-free family $\mathcal{F}$ is partitionable.

Consider a fork-tree family $\mathcal{F}$ that covers R but no proper subfamily of $\mathcal{F}$ covers R. Consider then problem C_1 with each cost $c(S, T) = 1$ for any $S \subseteq T \in \mathcal{F}$. We call a subtree S of a tree $T \in \mathcal{F}$ a *block* if $x(S, T) = 1$ in the given minimum value partition of R. In the given minimum value partition each block comes from a subtree $T \in \mathcal{F}$ and this subtree T is thought to be fixed together with the block.

If the theorem is true then the value of C_1 is just $|\mathcal{F}|$. We argue by contradiction so we take a family $\mathcal{F}$ so that the number of blocks in the minimum value partition is larger than $|\mathcal{F}|$. For this $\mathcal{F}$, any minimum value partition contains two blocks $S^1(T)$ and $S^2(T)$ coming from the same tree $T \in \mathcal{F}$. Denote by $S^1(T_0)$ and $S^2(T_0)$ two such blocks whose distance apart is minimal. Consider now that mi-

nimum value partition of R for which this minimal distance (between $S^1(T_0)$ and $S^2(T_0)$) is minimum.

This minimum distance is realized by a path (in R or in T_0) $P=\{v_0, v_1, \ldots, v_n\}$ where $v_0 \in S^1(T_0)$, $v_n \in S^2(T_0)$. Clearly $n>1$ as otherwise the block $S^1(T_0) \cup S^2(T_0)$ could replace the two blocks $S^1(T_0)$, $S^2(T_0)$ and the value of this partition would be less.

The point v_1 is covered by a block $S(T_1)$ coming from some $T_1 \in \mathscr{F}$. T_1 is different from T_0 for otherwise the distance between $S(T_1)$ and $S^2(T_0)$ would be less than that between $S^1(T_0)$ and $S^2(T_0)$.

Claim 1. $S^1(T_0) \not\subseteq T_1$, $S^2(T_0) \not\subseteq T_1$, $S(T_1) \not\subseteq T_0$.

Proof. If, for instance, $S^1(T_0) \subseteq T_1$, then $S^1(T_0) \cup S(T_1)$ would be a block of T_1 in a partition of R with smaller value. The other two cases are similar and left to the reader. ∎

Denote now by $B^1, \ldots, B^p$ the branches of $S(T_1)$ that stem from v_1 but do not contain v_2 and set $B^0 = S(T_1) \setminus (\{v_1\} \cup B^1 \cup \ldots \cup B^p)$. ($B_0$ may be empty.) Clearly $v_1 \in T_0$ and by Claim 1, $B^i \subseteq T_0$ cannot hold for every $i=0, 1, \ldots, p$.

Claim 2. *There is exactly one* $i \in \{0, \ldots, p\}$ *with* $B^i \not\subseteq T_0$.

Proof. Assume, on the contrary, that $B^i \not\subseteq T_0$ and $B^j \not\subseteq T_0$ and choose a point $x_1 \in B^i \setminus T_0$, $x_2 \in B^j \setminus T_0$. By Claim 1 there are points $y_1 \in S^1(T_0) \setminus T_1$ and $y_2 \in S^2(T_0) \setminus T_0$. Then the two trees $T_0, T_1 \in \mathscr{F}$ have a fork because the paths $[x_1, x_2]$ and $[y_1, y_2]$ meet at v_1. ∎

Now we are going to construct two new blocks from $S^1(T_0) \cup S(T_1)$ which will form (together with the other blocks) a minimum value partition of R with minimal distance less than n. Let B^i be the unique branch with $B^i \not\subseteq T_0$. Then $(S^1(T_0) \cup S(T_1)) \setminus B^i$ and B^i are two new blocks, the first coming from T_0 and the second from T_1. The distance from $S^2(T_0)$ to the new block $(S^1(T_0) \cup S(T_1)) \setminus B^i$ is less than n. This is a contradiction. ∎

5. Further observations

A 0-1 matrix A is called a *tree-matrix* if it is the node versus subtree incidence matrix of a subtree family of a tree. A *clique* in a graph is the vertex set of a maximal complete subgraph.

Proposition 6. *A is a tree-matrix if and only if A is the clique-node incidence matrix of a triangulated graph, including possibly some dominated rows.* ∎

This is easily proved using the Helly property of the subtrees of a tree and a theorem of Gavril [3] saying that the intersection graphs of subtrees of a tree are exactly the triangulated graphs. The proof is left to the reader.

It follows that we can consider the primal and dual variables for P_1 as node packing and clique weights in the corresponding triangulated graphs. If one checks now how our algorithm for P_1 works on the corresponding triangulated graph,

one can see that it does essentially the same thing as a perfect elimination scheme (see Golumbic [4]). In this sense what is new in Theorem 1 is the replacement of $a(v) \equiv 1$ by a monotone function.

As triangulated graphs can be recognised by a polynomial time algorithm it is of no surprise that the same is true for tree-matrices.

An Algorithm to Recognise Subtree Matrices

Let M be a 0-1 matrix

a) If M has a unit column e_j, set $M \leftarrow M \setminus e_j$.

b) If M has no unit columns and $m_{tj} \leqq m_{sj}$ for all j, set $M \leftarrow M \setminus m_t$ (drop row t from M)

c) If M has no unit columns and the rows form a clutter, stop. M is not a tree matrix.

d) Repeat steps a), b), c) until $M=0$, then M is a tree matrix.

The validation of the algorithm is left to the reader, we only prove c): any tree representation of M must have a leaf t. Either there exists a tree $T=\{t\}$ giving rise to a unit column or every tree containing t contains its neighbour s. In other words $m_{sj} \geqq m_{tj}$ for all j where s is the neighbour of t.

We now consider the covering problem C_0. In general it is NP-hard as any 0-1 covering problem reduces to it by adding a row of 1's. Therefore it is not surprising that its linear programming relaxation does not in general have a 0-1 solution. The simplest example showing this is the tree R depicted in Figure 2.

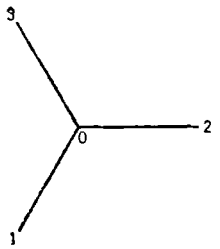

Fig. 2. $\mathcal{F}=\{T_1, T_2, T_3\}$, $T_1=\{0, 1, 2\}$, $T_2=\{0, 2, 3\}$, $T_3=\{0, 3, 1\}$, $c(T_1)=c(T_2)=c(T_3)=1$

Given that C_0 is well-solved for fork-free families, one might ask whether in such cases the linear programming relaxation has a 0-1 solution. But the three subtrees appearing in Figure 2 are fork-free, so this is not the case. However it is known that the associated linear program has an integer solution when dealing with distance subtrees $T_u=\{v \in V: d(v, u) \leqq \alpha\}$, and more generally when the 0-1 incidence matrix of the subtrees is totally balanced (see Kolen [7], Hoffman, Kolen, Sakarovitch [6] for details). It is readily shown using for instance the above algorithm that every totally balanced matrix is a subtree matrix.

Now both totally balanced (TB) matrices and fork-free (FF) families give rise to subtree matrices (SM). Are (TB) and (FF) related? The example of Figure 2

is forkfree, but cannot be totally balanced as the LP relaxation is noninteger. On the other hand, there exists a totally balanced matrix, for which there exists a unique representation as a family of subtrees and the family is not forkfree. Finally it is readily checked that the distance subtrees defined above are forkfree, so $FF \cap TB \neq \emptyset$.

Finally we return to problem P_2. We consider the economic lot sizing problem with backlogging, which was one of the problems motivating this research. The problem can be formulated as a simple plant location problem where the plants are located on an interval, (or in the terminology of this paper: at the vertices of a tree R having vertex set $V=\{1, \ldots, n\}$ and edges $(i, i+1)$, $i=1, \ldots, n-1$, so the tree is a simple path).

The formulation is:

$$\text{Min} \sum_{u=1}^{n} \sum_{v=1}^{n} c(u, v)x(u, v)$$

(ELS)
$$\sum_{u=1}^{n} x(u, v) = 1 \quad (v \in \{1, \ldots, n\})$$

$$x(u, v) - x(u, u) \leqq 0 \quad (u, v \in \{1, \ldots, n\}, \quad u \neq v)$$

$$x(u, v) \geqq 0 \quad (u, v \in \{1, \ldots, n\}),$$

where $x(u, v)$ is the fraction of the demand in period v produced in period u, $c(u, u)$ is the fixed cost of setting up in period u plus $p_u d_u$, $c(u, v)=p_u+c_u^+ + \ldots + c_{v-1}^+)d_v$ if $u>v$, $c(u, v)=(p_u+c_u^- + \ldots + c_{v+1}^-)d_v$ if $v<u$, where p_u is the price of unit production in period u, $c_u^+ \geqq 0$ and $c_u^- \geqq 0$ are the price of unit transportation from u to $u+1$ and $u+1$ to u respectively and d_v is the demand in period v. It is typically imposed that $x(u, u) \in \{0, 1\}$. Now we can show

Theorem 7. ELS *has an optimal* 0-1 *solution.*

Outline of proof. It is easily shown that optimal solutions to ELS satisfy the inequalities

$$x(u, u) \geqq x(u, u+1) \geqq x(u, u+1) \geqq \ldots$$

and

$$x(u, u) \geqq x(u, u-1) \geqq x(u, u-2) \geqq \ldots, \quad \text{for all} \quad u \in \{1, \ldots, n\}.$$

Adding these inequalities to ELS, and removing the dominated inequalities, the resulting linear program is of the form P_2, with R an interval graph, $a(v)=1$ for $v \in V$, and equality constraints at each node. It follows from Theorem 3 that ELS has an optimal integer solution. ∎

References

[1] I. BÁRÁNY, T. J. VAN ROY, and L. A. WOLSEY, Uncapacitated Lot-Sizing: The Convex Hull of Solutions, *Mathematical Programming Study*, **22** (1984), 32—43.
[2] A. FRANK, Some Polynomial Algorithms for Certain Graphs and Hypergraphs, *Proceedings of the 5th British Combinatorial Conference, Congressus Numerantium* **XV**, *Utilitas Math.*, Winnipeg (1976).
[3] F. GAVRIL, The intersection Graphs of Subtrees in Trees are Exactly the Chordal Graphs, *Journal of Combinatorial Theory, B* **16**, (1974), 47—56.
[4] M. C. GOLUMBIC, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, (1980).
[5] H. GRÖFLIN and T. M. LIEBLING, Connected and Alternating Vectors: Polyhedra and Algorithms, *Mathematical Programming*, **20** (1981), 233—244.
[6] A. J. HOFFMAN, A. W. J. KOLEN and M. SAKAROVITCH, Totally Balanced and Greedy Matrices, *Prepring BW* 165/82, *Stiching Mathematisch Centrum*, Amsterdam, (1982).
[7] A. W. J. KOLEN, Solving Covering Problems and the Uncapacitated Plant Location Problem on Trees, *Preprint BW* 163/82, *Stichting Mathematisch Centrum*, Amsterdam (1982).

I. Bárány

Mathematical Institute of the Hungarian Academy of Sciences
Reáltanoda u. 13—15.
1053, Hungary

L. A. Wolsey

Center Op. Res, Econometrics
Université Catholique de Louvain
1348 Louvain-la-Neuve
Belgium

J. Edmonds

Dept. of Combinatorics and Optimization
University of Waterloo
Ontario, Canada