Top (2008) 16: 114–125
DOI 10.1007/s11750-007-0037-9

ORIGINAL PAPER

Improved algorithms for the multicut and multiflow problems in rooted trees

A. Tamir

Received: 6 May 2007 / Accepted: 15 November 2007 / Published online: 4 December 2007

Abstract Costa et al. (Oper. Res. Lett. 31:21–27, 2003) presented a quadratic $O(\min(Kn, n^2))$ greedy algorithm to solve the integer multicut and multiflow problems in a rooted tree. (n is the number of nodes of the tree, and K is the number of commodities). Their algorithm is a special case of the greedy type algorithm of Kolen (Location problems on trees and in the rectilinear plane. Ph.D. dissertation, 1982) to solve weighted covering and packing problems defined by general totally balanced (greedy) matrices. In this communication we improve the complexity bound in Costa et al. (Oper. Res. Lett. 31:21–27, 2003) and show that in the case of the integer multicut and multiflow problems in a rooted tree the greedy algorithm of Kolen can be implemented in subquadratic $O(K + n + \min(K, n)\log n)$ time. The improvement is obtained by identifying additional properties of this model which lead to a subquadratic transformation to greedy form and using more sophisticated data structures.

Keywords Maximum integral multiflows · Minimum multicuts · Totally balanced matrices · Greedy matrices · Rooted trees

Mathematics Subject Classification (2000) 90C27 · 90C10

1 Introduction

Let $T = (V, E)$ be a tree with node set $V = \{v_1, \dots, v_n\}$ and edge set E. Each edge $e \in E$ has a positive integer capacity (weight) u_e. Let $\{(s_k, t_k) : s_k \neq t_k, k = 1, \dots, K\}$ be a list of K distinct pairs of nodes. Associate a commodity k with the pair (s_k, t_k)

A. Tamir (✉)
School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel
e-mail: atamir@post.tau.ac.il

and designate s_k as the source and t_k as the sink of this commodity. The integral multiflow problem is to maximize the sum of the integral simultaneous flows corresponding to the K commodities, subject to the capacity and flow conservation constraints. The respective multicut problem is to find a minimum weight set of edges whose removal separates each pair (s_k, t_k), $k = 1, \ldots, K$.

Consider the formulation of the integral multiflow problem as a packing problem (Costa et al. 2003). For $k = 1, \ldots, K$, let p_k be the unique path from s_k to t_k, and let f_k denote the integral flow on p_k. For $k = 1, \ldots, K$ and $e \in E$, define $a_{ek} = 1$ if $e \in p_k$ and $a_{ek} = 0$ if $e \notin p_k$. Let $\mathbb{N}$ denote the set of all nonnegative integer numbers. The problem is to find

$$\max \sum_{k=1}^{K} f_k,$$

subject to

$$\begin{aligned} &\sum_{k=1}^{K} a_{ek} f_k \le u_e, \quad \forall e \in E, \\ &f_k \in \mathbb{N}, \quad \forall k = 1, \ldots, K. \end{aligned} \tag{1}$$

Similarly, the formulation of the integral multicut problem as a covering problem is

$$\min \sum_{e \in E} u_e x_e,$$

subject to

$$\begin{aligned} &\sum_{e \in E} a_{ek} x_e \ge 1, \quad \forall k = 1, \ldots, K, \\ &x_e \in \{0, 1\}, \quad \forall e \in E. \end{aligned} \tag{2}$$

For general undirected trees, both the multicut and the integral multiflow problems are NP-hard, as shown respectively by Kolen and Tamir (1990) and Garg et al. (1997). When T is directed and for each $k = 1, \ldots, K$, the path p_k is directed from s_k to t_k, using total unimodularity properties, the above pair of packing-covering integer programs have been shown to be a pair of primal-dual linear programs (see Costa et al. 2003 and the references therein). They are both polynomially solvable by using standard network flow algorithms.

Costa et al. (2003) focused on the case where the tree T is rooted. In this case there is a distinguished root, say v_1, and the edges are directed so that there is a unique directed path from v_1 to any other node. In this model, for each $k = 1, \ldots, K$, s_k is an ancestor of t_k on the rooted tree, and p_k is a directed path from the source s_k to the sink t_k. They present a greedy procedure to find an optimal multiflow and use duality properties to obtain an optimal multicut. In fact, their algorithm is a specialization of the quadratic time greedy type algorithm of Kolen (1982) (see also Kolen 1986;

Hoffman et al. 1985; Kolen and Tamir 1990) to solve weighted covering and packing problems defined by general totally balanced and greedy matrices. (Exact definitions of these and related concepts are given in the next section.) The complexity of the algorithm in Costa et al. (2003) is $O(\min(Kn, n^2))$.

In this note we also concentrate on the multicut and multiflow problems in rooted trees. Our main objective is to show that this model can be solved in subquadratic time. We identify additional properties of this model which lead to a subquadratic transformation into greedy form, and apply more sophisticated data structures, to show that the above greedy algorithm of Kolen can be implemented in subquadratic $O(K + n + \min(K, n) \log n)$ time. Since $K \leq n(n-1)/2$ for any rooted tree, we conclude that our modified complexity bound coincides with the quadratic bound in Costa et al. (2003) if and only if $K = O(1)$ or $K = \theta(n^2)$. Our modified bound strictly improves upon the bound in Costa et al. (2003) in all other cases. For example, when $K = \theta(n^\delta)$, the improvement is from $O(n^{1+\delta})$ to $O(n)$ if $0 < \delta < 1$, from $O(n^2)$ to $O(n \log n)$ if $\delta = 1$, and from $O(n^2)$ to $O(n^\delta)$ if $1 < \delta < 2$.

In the next section we identify specific properties of the totally balanced matrix defining the above multiflow and multicut problems for rooted trees. We then present the subquadratic algorithm transforming this matrix into greedy form. In Sects. 3 and 4 we describe the algorithms for the multiflow and multicut problems, respectively. Section 5 is devoted to some related flow and cut problems.

2 Totally balanced and greedy matrices

To facilitate the discussion we first recall several relevant general definitions and results from Hoffman et al. (1985), Kolen and Tamir (1990).

Let A be an $n \times m$ $\{0, 1\}$ matrix. A is *balanced* if it does not contain a square submatrix of odd size with row and column sums equal to 2. A is *totally balanced* if it does not contain a square submatrix with row and column sums equal to 2 and no identical columns. (Note that a $\{0, 1\}$ totally unimodular matrix is balanced but not necessarily totally balanced.)

A has the *nest ordering property for columns* if for $i = 1, \ldots, n$, the following holds: the supports of all columns containing a 1 in row i can be totally ordered by inclusion when they are restricted to rows with index greater than or equal to i.

A is in *standard greedy form* (greedy, in short) if it does not contain a 2×2 submatrix of the form

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

Given a pair of $\{0, 1\}$ vectors $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ in $\mathbb{R}^n$, we say that x is *lexical* larger than y if $x \neq y$, and there is an index $i = 1, \ldots, n$ such that $x_i = 1$, $y_i = 0$, and $x_q = y_q$ for all $q = i + 1, \ldots, n$. We say that a set of $\{0, 1\}$ vectors $\{x^1, \ldots, x^m\}$ in $\mathbb{R}^n$ is in *lexical nondecreasing order* if for any $j = 1, \ldots, m - 1$, $x^j = x^{j+1}$ or x^{j+1} is lexical larger than x^j.

The following results on general totally balanced matrices are proven in Hoffman et al. (1985), Kolen and Tamir (1990).

Theorem 2.1 *If A is a greedy matrix, it is totally balanced.*

If A is totally balanced, there is a permutation of its rows such that the permuted matrix has the nest ordering property for columns.

If A has the nest ordering property for columns and if its columns are ordered in a lexical nondecreasing order, then A is in greedy form.

The $(n-1)\times K$ edge-path incidence matrix $A=(a_{ek})$, defining the above integral multiflow and multicut problems for the rooted tree case, was shown to be totally balanced by Frank (1977). See Tamir (1987) for generalizations of the results in Frank (1977). (Note that Frank considers only a node-path incidence matrix. However, an edge can be represented by an auxiliary node located at its midpoint. Hence, an edge-path incidence matrix can be converted to a node-path matrix in case of trees.) In particular, the integrality constraints on the flows f_k in the above formulation can be replaced by nonnegativity constraints.

When a totally balanced matrix is in greedy form, the packing problem can be solved by the greedy algorithm in Hoffman et al. (1985), Kolen (1982, 1986), Kolen and Tamir (1990). (The solution to the dual covering problem can then easily be derived.) Thus, our first computational task is to transform a given totally balanced matrix into greedy form. By the above theorem, it is sufficient to find a row permutation which induces the nest ordering property and then permute the columns in a lexical nondecreasing order.

The most efficient algorithms to transform a general totally balanced matrix into standard greedy form appear in Lubiw (1987), Paige and Tarjan (1987), Spinard (1993). All these algorithms, as well as the original algorithm in Hoffman et al. (1985), assume that the matrix A is given explicitly. As a result, in the worst case, their complexity is bounded below by $\Omega(nm)$, where n and m are the number of rows and columns of A, respectively. The implementation of the greedy algorithm will then take $O(nm)$ extra time. Hence, to obtain subquadratic time algorithms for special cases like the multicut and multiflow problems in rooted trees, discussed in this note, we first need to have more efficient subquadratic algorithms for the transformation.

We will show that in our case, where A is the incidence matrix of a family of m directed paths of a rooted tree versus the edges of the tree, the row and column permutations transforming the matrix into greedy form can be obtained in $O(n+m\log m)$ time. This is the crucial part of our procedure to solve problems (1, 2), and it will then lead to an improved $O(K+n+\min(K,n)\log n)$ algorithm for the integer multiflow and multicut problems on rooted trees.

2.1 Permuting the incidence matrix of a family of paths in a rooted tree into greedy form

The transformation into greedy form is based on the following theorem.

Theorem 2.2 *Let T be a rooted tree, and let $A=(a_{ek})$ be the $(n-1)\times K$ edge-path incidence edge-path incidence matrix of a family of K distinct directed paths on T. There is an $O(n+K\log K)$ time algorithm to obtain the row and column* *permutations, transforming A to greedy form. If T is a rooted path, the running time reduces to $O(n+K)$.*

Proof Assume that the rooted tree T is embedded in the plane. Starting with the root v_1, reindex (label) the nodes by their depth (level), where in each level nodes are indexed consecutively, following the left right order. If v_1 has i children (these are the nodes of depth 1), they are reindexed as $v_2, \dots, v_{i+1}$, from "left to right." If there are j nodes of depth 2, they will then be reindexed as $v_{i+2}, \dots, v_{i+1+j}$, etc. Each edge of T is now assigned the index (label) of its deeper node.

We define the row permutation by decreasingly ordered the edges (rows) by their label. The first edge according to this ordering is the one incident to the deepest rightmost leaf node v_n. If v_{n-1} is a leaf, the second edge (row) is the one incident to v_{n-1}, otherwise, the second edge is the unique edge having v_{n-1} as its deeper node, etc. The last edge (row) is the unique edge having v_2 as its deeper node. We show that the nest ordering property for columns is achieved with this row permutation.

Consider the ith edge (row) according to the above ordering. It is incident to node v_{n+1-i}. (Note that each edge which is a descendant of v_{n+1-i} is some jth edge (row) with $j < i$.) Let $P(v_1, v_{n+1-i})$ denote the path connecting v_{n+1-i} with the root v_1. Let p_k and p_u be two directed paths containing the ith edge. Then, either $p_k \cap P(v_1, v_{n+1-i}) \subseteq p_u \cap P(v_1, v_{n+1-i})$ or $p_u \cap P(v_1, v_{n+1-i}) \subseteq p_k \cap P(v_1, v_{n+1-i})$. This proves the nest ordering property of A. The effort to obtain the above row permutation is clearly $O(n)$.

Next, we need to permute the columns in lexical nondecreasing order. Consider a pair of distinct paths (s_k, t_k) and (s_q, t_q). If the label of the node s_k is smaller than that of the node s_q, the column of the matrix corresponding to the path (s_k, t_k) is lexical larger than the respective column of (s_q, t_q). Hence, suppose that $s_k = s_q$. Let v_j be the lowest common ancestor of the nodes t_k and t_q. If $t_q = v_j$, then the path (s_q, t_q) is a subpath of (s_k, t_k), and the column of the matrix corresponding to the path (s_k, t_k) is lexical larger than the respective column of (s_q, t_q). Hence, suppose that both t_k and t_q are proper descendants of v_j. Let $v_{k(j)}$ and $v_{q(j)}$ be the two children of v_j on the paths connecting v_j to t_k and t_q, respectively. The column of the matrix corresponding to the path (s_k, t_k) is lexical larger than the respective column of (s_q, t_q) if and only if $k(j) < q(j)$.

We claim that after spending $O(n)$ effort on preprocessing, it takes constant time to compare two columns (paths) by the lexical ordering. First, to find v_j, the lowest common ancestor of a pair of nodes t_k and t_q, we can use the data structures in Harel (1980), Harel and Tarjan (1984) (see also Bender and Farach-Colton 2000, Berkman and Vishkin 1993). Next, to find the two children of v_j, $v_{k(j)}$ and $v_{q(j)}$, we can use the data structure in Berkman and Vishkin (1994).

With the above machinery it will take $O(n + K \log K)$ time to sort the columns of the matrix and obtain the nondecreasing lexical ordering. Observe that $K \leq n(n-1)/2$.

We note that in the special case where T itself is a rooted path, the transformation to greedy form takes $O(n+K)$ time. Given a pair of distinct paths (s_k, t_k) and (s_q, t_q), the former is lexical larger than the latter if and only if the label of s_k is smaller than that of s_q, or $s_k = s_q$ and the label of t_k is larger than that of t_q. Hence, the nondecreasing lexical ordering of the columns can be obtained in $O(n+K)$ time by using a standard radix sort procedure. This completes the proof. □

3 Solving the multiflow problem on a rooted tree

We consider the following generalization of the multiflow problem on a rooted tree, where we introduce integer upper bounds $\{b_k\}$ on the flow variables $\{f_k\}$. We refer to this generalization as the *bounded multiflow problem*,

$$\max \sum_{k=1}^{K} f_k,$$

subject to

$$\begin{aligned} &\sum_{k=1}^{K} a_{ek} f_k \le u_e, \quad \forall e \in E, \\ &0 \le f_k \le b_k, \quad f_k \in \mathbb{N}, \ \forall k = 1, \dots, K. \end{aligned} \tag{3}$$

For $k=1,\dots,K$, let a^k denote the kth column of A. Suppose that the matrix $A=(a_{ek})$ is already in standard greedy form, i.e., for $k=1,\dots,K-1$, either $a^{k+1}=a^k$ or a^{k+1} is lexical larger than a^k.

From Hoffman et al. (1985), Kolen (1982, 1986), Kolen and Tamir (1990) we conclude that an optimal solution to the above packing problem is defined by the following recursive greedy scheme:

$$f_1 = \min\Big\{b_1; \min_{\{e:a_{e1}=1\}} u_e\Big\}$$

and for $k=2,\dots,K$,

$$f_k = \min\left\{b_k; \min_{\{e:a_{ek}=1\}} \left\{u_e - \sum_{j=1}^{k-1} a_{ej} f_j\right\}\right\}.$$

We now show that this algorithm can be implemented in $O(n + K\log n)$ time.

Throughout the algorithm, each edge $e \in E$ of the rooted tree is associated with a nonnegative integer u'_e reflecting the residual remaining capacity of the edge. Initially $u'_e = u_e$.

The Multiflow Greedy Algorithm:

For $k=1,\dots,K$,

Step 1. Compute $c^k = \min_{e\in p_k} u'_e$. Set $f_k = \min\{b_k, c^k\}$.
Step 2. For each $e \in p_k$, subtract f_k from u'_e.

Theorem 3.1 *The integer bounded multiflow problem* (3) *defined on a rooted tree T is solvable in $O(n+K\log n)$ time.*

Proof In the first phase we use the result stated in Theorem 2.2 and find the row and column permutations converting the matrix into greedy form. The computational effort of this phase is $O(n + K \log K)$. Since $K \leq n(n-1)/2$, the latter bound is $O(n + K \log n)$.

Next we apply the above Multiflow Greedy Algorithm. In the first step we compute the minimum of the residual capacities of the edges along a given directed path, while in the second step we subtract some constant from these residual capacities. Hence, to implement this greedy algorithm we can use the dynamic trees data structures in Sleator and Tarjan (1983, 1985), Tarjan (1997). (In particular, see page 677 in Sleator and Tarjan 1985.) With these data structures, after spending $O(n)$ time on preprocessing, each one of the K iterations of the algorithm can be implemented in $O(\log n)$ time. Thus, the total time to solve the integer multiflow problem with upper bounds on rooted trees is $O(n + K \log n)$. This completes the proof. □

An improved bound can be attained for problem (1), where no explicit upper bounds on $\{f_k\}$ are stated (this is the model in Costa et al. 2003).

Theorem 3.2 *The integer multiflow problem* (1) *defined on a rooted tree T is solvable in $O(K + n + \min(K, n) \log n)$ time.*

Proof If $K < n$, the result follows directly from the previous theorem. Suppose that $K \geq n$. In this case we perform a preprocessing phase which reduces the multiflow problem into an equivalent problem with at most n commodities (directed paths). This phase takes $O(K + n)$ time. Specifically, we show that in the equivalent problem for each node v_i, there is at most one path p_k such that $t_k = v_i$.

Let $K(i) = \{k \in \{1, \dots, K\} : t_k = v_i\}$ denote the index set of all directed paths having v_i as their sink. Let $p_{k(i)}$ denote the shortest path in $\{p_k : k \in K(i)\}$. Consider $\{f_k : k = 1, \dots, K\}$, an optimal solution to the multiflow problem (1). Then it is easy to see that the multiflow $\{f'_k : k = 1, \dots, K\}$ defined by $f'_k = f_k$ for $k \notin K(i)$, $f'_k = 0$ for $k \in K(i)$, $k \neq k(i)$, and $f'_{k(i)} = \sum_{k \in K(i)} f_k$ is also optimal.

From the above it follows that for each node v_i we can omit all paths in $\{p_k : k \in K(i)\}$, except the shortest one, $p_{k(i)}$. The effort needed for this phase is clearly $O(K + n)$. After the elimination phase we are left with $K' < n$ paths.

Finally, we apply the previous theorem to the equivalent multiflow problem with K' directed paths. Hence, when $K \geq n$, the total effort to solve problem (1) is $O(K + n \log n)$. This completes the proof. □

We observe that to apply the above greedy algorithm we do not need to obtain the complete lexical ordering of the columns and transform the incidence matrix into greedy form. The following partial ordering will suffice. After the reindexing of the nodes, find a node v_j with the highest index such that there exists a path p_k with $s_k = v_j$. Let $P^j = \{p_q : s_q = v_j\}$. Apply Step 1 and Step 2 of the greedy algorithm to the paths in P^j in arbitrary order. Continue with a node, say v_m, with the second highest index such that there exists a path p_k with $s_k = v_m$, etc. (Note that the $O(\min(Kn, n^2))$ algorithm in Costa et al. (2003) to solve the multiflow problem (1) uses only the above partial ordering).

In spite of the fact that we do not need the complete lexical ordering, the overall complexity of the greedy algorithm will still be as stated in the above theorem.

The implementation of the greedy algorithm to a matrix with a complete lexical ordering of the columns is required in order to solve the dual multicut problem in the next section.

4 Solving the multicut problem on a rooted tree

To solve the multicut problem (2) defined in the introduction, we use the optimal solution $\{f_k\}$ of the multiflow problem (1) from the last section (with $b_k = \infty$ for $k = 1, \dots, K$) and apply the general algorithm in Hoffman et al. (1985), Kolen (1982, 1986), Kolen and Tamir (1990) (specifically, we follow the presentation on page 272 in Kolen and Tamir 1990). From the last section we assume that at this stage there are only $K' \leq \min(K, n)$ relevant paths left and we relabel them as $\{p_k : k = 1, \dots, K'\}$.

For $k = 1, \dots, K'$ and $e \in E$, if f_k is such that $\sum_{j=1}^{k-1} a_{ej} f_j < u_e$ and $\sum_{j=1}^{k} a_{ej} f_j = u_e$, we say that the edge e is *saturated* by f_k. An edge e is *binding* if $\sum_{j=1}^{K'} a_{ej} f_j = u_e$.

Let E'' be the subset of all binding edges, and let F'' be the index set of all flow variables $\{f_k\}$ which saturate an edge. By definition each edge in E'' is saturated by exactly one flow variable with index in F'', and each flow variable with index in F'' saturates at least one edge in E''. Both sets E'' and F'' can be generated during the implementation of the Multiflow Greedy Algorithm. An edge e is saturated by f_k if and only if, in Step 2 of the algorithm, $u'_e > 0$ and $u'_e - f_k = 0$.

Define $E^* \subseteq E''$ as follows (initiating with $E^* = \emptyset$):

Add the edge $e' \in E''$ with largest row index to E^* and delete all edges $e \in E''$ for which $a_{ek} = a_{e'k} = 1$ for some $k \in F''$. (Recall that the underlying matrix is in standard greedy form, and the rows (edges) and columns (paths) are properly indexed.)

Repeat until $E'' = \emptyset$.

It follows from Hoffman et al. (1985), Kolen (1982, 1986), Kolen and Tamir (1990) that an optimal solution to the multicut problem is given by setting $x_e = 1$ for $e \in E^*$ and $x_e = 0$ for $e \notin E^*$.

We show how to implement the above scheme to find E^* in $O(n)$ time.

Lemma 4.1 *Assume that the sets E'' and F'' are given. Then, the optimal solution to the multicut problem can be found in $O(n)$ time.*

Proof We mark the edges in E'' as *special* and the rest as *regular*. To each node v_j of the rooted tree T, we add a new leaf edge (v_j, v'_j) with a new leaf node v'_j. We label these leaf edges as special. Let T' denote the augmented tree with the $2n$ nodes, $\{v_1, \dots, v_n\}$ and $\{v'_1, \dots, v'_n\}$. There is a partial order induced on the set of special edges by the original rooted tree T. This partial order can be represented by a rooted tree, say T'', whose root is the node v'_1 or the edge (v'_1, v_1). Specifically, T'' is obtained from T' by contracting all the regular edges. (The edge set of T'' consists of E'' and of the n leaf edges (v_j, v'_j), $j = 1, \dots, n$.)

The implementation of the algorithm is as follows:

The Multicut Algorithm:

Step 1. Let E^+ be the subset of all edges in T'' which are children of the root edge (v_1, v_1'). Augment the edges in E^+ to E^*. Let T^+ be the subtree of T' induced by the root and E^+.

Step 2. Select a path p_k with s_k in T^+ and s_k not a leaf of T^+. If t_k is in T^+ or $k \notin F''$, discard p_k. Otherwise, consider the special leaf edge attached to the node t_k. Moving along the path in T'', from this leaf edge to the root of T'', contract all the (non-leaf) special edges along this path in T''. Also, in T' convert each one of the contracted edges to a regular status, i.e., delete it from E''. Discard p_k.

Step 3. If there is still a path p_k with s_k in T^+ and s_k not a leaf of T^+, go to Step 2. Otherwise, go to Step 4.

Step 4. Contract all the non-leaf edges of T^+ in T'. If T'' contains non-leaf edges, go to Step 1. Otherwise, stop. ($E'' = \emptyset$.)

Throughout the implementation of the algorithm, each special edge (with the exception of the special leaf edges that we have augmented) is visited and converted to a regular edge exactly once, and each path p_k is considered exactly once. For example, if a path p_k, $k \in F''$, is selected in some iteration of Step 2, and p_k contains m special edges, then the effort to contract these edges and delete them from E'' is $O(m)$. Therefore, the total effort spent to find E^* is $O(n + K')$. Since $K' \le \min(K, n)$, the total computational effort is $O(n)$. □

In the case where T itself is a directed path, there are recursive algorithms which solve the multicut problem directly and do not rely on a solution to the multiflow problem. See Hassin and Tamir (1991) for an $O(n + K')$ algorithm.

The next theorem summarizes the above algorithmic results.

Theorem 4.1 *The integer multicut problem* (2) *defined on a rooted tree T is solvable in $O(K + n + \min(K, n)\log n)$ time. If T is a rooted path, the running time reduces to $O(n + K)$.*

Proof We first solve the integer multiflow problem (1) and find the sets E'' and F''. From Theorem 3.2 the total effort of this phase is $O(K + n + \min(K, n)\log n)$. To find the solution to the integer multicut problem we then apply the $O(n)$ algorithm in the above lemma.

In the case where T itself is a directed path, there are recursive algorithms which solve the multicut problem directly and do not rely on a solution to the multiflow problem. See Hassin and Tamir (1991) for an $O(n + K)$ algorithm. □

5 Related problems

We note that in addition to the multicut and multiflow problems, there are other related optimization problems defined by the edge-path incidence matrix $A = (a_{ek})$ on rooted trees. We briefly discuss two such examples in the literature and note their relations to problems (1), (2).

5.1 The network improvement problem on a tree

In the network improvement problem the goal is to modify the lengths of the edges so that after the modification the distances from a root node to all other nodes are within given bounds, and the total modification cost is minimized. The problem is NP-hard on general graphs but polynomially solvable on rooted trees. See Zhang et al. (2004) and the references therein.

Using the above notation for rooted trees, the improvement problem can be formulated as the following weighted multicut (multicover) problem with $n-1$ paths, all rooted at v_1, the root of the underlying tree. Specifically, each node v_k, $k=2,\dots,n$, is associated with the path p_k connecting the root v_1 to v_k and with a nonnegative real number r_k representing the required decrement of the length of p_k. For each edge $e\in E$, c_e denotes the per unit cost of decrementing the length of e, and d_e denotes an upper bound on this decrement.

$$\min \sum_{e\in E} c_e x_e,$$

subject to

$$\begin{aligned}
&\sum_{e\in E} a_{ek} x_e \ge r_k, \qquad \forall k=2,\dots,n,\\
&0\le x_e\le d_e, \qquad \forall e\in E.
\end{aligned}$$

The above multicovering problem is solved in $O(n\log n)$ time (Zhang et al. 2004). We note that when $r_k=1$ for $k=2,\dots,n$ and $d_e=1$ for each $e\in E$, the above model reduces to a special case of problem (2) with $K=n-1$ and $(s_k,t_k)=(v_1,v_k)$ for $k=2,\dots,n$.

5.2 The 2-edge-connectivity augmentation problem

The second example is the minimum weighted covering problem of the edges of the rooted tree by paths considered in Conforti et al. (2004), Galluccio and Proietti (2003). This problem arises from 2-edge-connectivity augmentation problems. (In comparison, the multicut problem (2) asks for a minimum weighted covering of the paths by edges). Using the above notation, the problem is formulated as ($\{w_k\}$ are nonnegative)

$$\min \sum_{k=1}^{K} w_k z_k,$$

subject to

$$\begin{aligned}
&\sum_{k=1}^{K} a_{ek} z_k \ge 1, \qquad \forall e\in E,\\
&z_k\in\{0,1\}, \qquad \forall k=1,\dots,K.
\end{aligned}\tag{4}$$

Again, since the underlying matrix $A = (a_{ek})$ is totally balanced, the problem can be solved by Kolen's greedy algorithm. In fact, the algorithm in Galluccio and Proietti (2003) is also a special case of Kolen's general algorithm. Its implementation in Galluccio and Proietti (2003) has an $O(n + K\alpha(K, n)\alpha(K\alpha(K, n), n))$ complexity. (α is the inverse of the Ackermann's function Sharir and Agarwal 1995). In spite of the similarity in formulation between problem (4) of covering edges by paths and the multicut problem (2), in which we cover paths by edges, it is not yet clear whether the solution procedures in Galluccio and Proietti (2003) for the former are also applicable to solve the latter.

In the case where T itself is a rooted path, the complexity of the algorithm in Galluccio and Proietti (2003) to solve the above augmentation problem reduces to $O(n + K\alpha(K, n))$. In this case the incidence matrix has the column consecutive 1's property, i.e., in each column, the 1's appear consecutively. The dual of the augmentation problem (4) is the following packing problem of edges into paths:

$$\max \sum_{e \in E} x_e,$$

subject to

$$\sum_{e \in E} a_{ek} x_e \le w_k, \quad \forall k = 1, \dots, K,$$
$$x_e \ge 0, \quad \forall e \in E.$$

The constraints of this packing problem have the row consecutive 1's property. Therefore, the problem can be solved as a shortest path problem on a directed network with n nodes and K edges (Hochbaum and Levin 2006). Since the edge lengths $\{w_k\}$ are nonnegative, the time to solve the problem is $O(K + n \log n)$. The latter bound improves upon the $O(n + K\alpha(K, n))$ bound in Galluccio and Proietti (2003), whenever $K = \Omega(n \log n)$.

5.3 Open problems

We have shown in Sect. 2.1 how to find the row and column permutations which transform the edge-path incidence matrix on a rooted tree into greedy form in $O(n + K \log K)$ time. It is not known whether this complexity bound is optimal.

It is an open question whether the $O(n + K \log n)$ complexity bound for the greedy algorithm in Sect. 3 is best possible for solving the integer bounded multiflow problem (3) on rooted trees. Our implementation relies on data structures that can execute Step 1 and Step 2 in $O(\log n)$ time for each given value of k. We note that the data structures described in Alon and Schieber (1989), Chazelle and Rosenberg (1991), Tarjan (1978) can execute Step 1 in $O(\alpha(n))$. We are unaware of any data structure that can perform both Step 1 and Step 2 in $O(\alpha(n))$ time or even $o(\log n)$ time, after spending $O(n)$ effort on preprocessing.

References

Alon N, Schieber B (1989) Optimal preprocessing for answering on-line product queries. Technical Report, Tel Aviv University

Bender MA, Farach-Colton M (2000) The LCA problem revisited. In: Proceedings of the 4-th Latin American theoretical informatics symposium (LATIN), pp 88–94

Berkman O, Vishkin U (1993) Recursive star-tree parallel data structure. SIAM J Comput 22:221–242

Berkman O, Vishkin U (1994) Finding level-ancestors in tree. J Comput Syst Sci 48:214–230

Chazelle B, Rosenberg B (1991) The complexity of computing partial sums off-line. Int J Comput Geom Appl 1:33–45

Conforti M, Galluccio A, Proietti G (2004) Edge-Connectivity Augmentation and Network Matrices. In: Hrmokovic J, Nagi M, Westfechtel B (eds) Graph-theoretic concepts in computer science, 30-th international workshop 2004, Germany. LNCS, vol 3353. Springer, Berlin, pp 355–364

Costa M-C, Letocart L, Roupin F (2003) A greedy algorithm for multicut and integral multiflow in rooted trees. Oper Res Lett 31:21–27. See Erratum, Oper Res Lett 34:477 (2006)

Frank A (1977) On a class of balanced hypergraphs. Discrete Math 20:11–20

Galluccio A, Proietti G (2003) Polynomial time algorithms for 2-edge connectivity augmentation problems. Algorithmica 36:361–374

Garg N, Vazirani VV, Yannakakis M (1997) Primal-dual approximation algorithms for integral flow and multicut in trees. Algorithmica 18:3–20

Harel D (1980) A linear time algorithm for the lowest common ancestors problems. In: Proceedings of the 21-st IEEE symposium on foundations of computer science (FOCS), pp 308–319

Harel D, Tarjan RE (1984) Fast algorithms for finding nearest common ancestors. SIAM J Comput 13:338–355

Hassin R, Tamir A (1991) Improved complexity bounds for location problems on the real line. Oper Res Lett 10:395–402

Hochbaum DS, Levin A (2006) Optimizing over consecutive 1's and circular 1's constraints. SIAM J Optim 17:311–330

Hoffman AJ, Kolen A, Sakarovitch M (1985) Totally-balanced and greedy matrices. SIAM J Algebraic Discrete Methods 6:721–730

Kolen A (1982) Location problems on trees and in the rectilinear plane. PhD dissertation, Mathematisch Centrum, Amsterdam

Kolen A (1986) Tree network and planar rectilinear location theory. CWI tract 25, Mathematisch Centrum, Amsterdam

Kolen A, Tamir A (1990) Covering problems. In: Mirchandani PB, Francis RL (eds) Discrete location theory. Wiley, New York

Lubiw A (1987) Doubly lexical ordering of matrices. SIAM J Comput 16:854–879

Paige R, Tarjan RE (1987) Three partition refinement algorithms. SIAM J Comput 16:973–989

Sleator D, Tarjan RE (1983) A data structure for dynamic trees. J Comput Syst Sci 26:362–391

Sleator D, Tarjan RE (1985) Self-adjusting binary trees. J ACM 32:652–685

Spinard JP (1993) Doubly lexical ordering of dense 0–1 matrices. Inf Process Lett 45:229–235

Sharir M, Agarwal PK (1995) Davenport–Schinzel sequences and their geometric applications. Cambridge University Press, Cambridge

Tamir A (1987) Totally balanced and totally unimodular matrices defined by center location problems. Discrete Appl Math 16:245–263

Tarjan RE (1978) Complexity of monotone networks for computing conjunctions. Ann Discrete Math 2:121–133

Tarjan RE (1997) Dynamic trees as search trees via Euler tours, applied to the network simplex algorithm. Math Program 78:167–177

Zhang JZ, Yang XG, Cai MC (2004) Inapproximability and a polynomially solvable special case of a network improvement problem. Eur J Oper Res 155:251–257