Theory and Methodology

Minimal spanning trees with a constraint on the number of leaves

Lucinda Matos Fernandes *, Luis Gouveia **

*Dep. Matemática – Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Quinta da Torre, 2825 Monte da Caparica, Portugal

**DEIO-CIO, Faculdade de Ciências, Universidade de Lisboa, Bloco C / 2 – Campo Grande, Cidade Universitaria, 1700 Lisboa, Portugal

Received 30 August 1995; accepted 15 August 1996

Abstract

In this paper we discuss minimal spanning trees with a constraint on the number of leaves. Tree topologies appear when designing centralized terminal networks. The constraint on the number of leaves arises because the software and hardware associated to each terminal differs accordingly with its position in the tree. Usually, the software and hardware associated to a “degree-1” terminal is cheaper than the software and hardware used in the remaining terminals because for any intermediate terminal \( j \) one needs to check if the arrival message is destined to that node or to any other node located after node \( j \). As a consequence, that particular terminal needs software and hardware for message routing. On the other hand, such equipment is not needed in “degree-1” terminals. Assuming that the hardware and software for message routing in the nodes is already available, the above discussion motivates a constraint stating that a tree solution has to contain exactly a certain number of “degree-1” terminals. We present two different formulations for this problem and some lower bounding schemes derived from them. We discuss a simple local-exchange heuristic and present computational results taken from a set of complete graphs with up to 40 nodes. Integer Linear Programming formulations for related problems are also discussed at the end. © 1998 Elsevier Science B.V.

Keywords: Integer linear programming; Linear programming relaxations; Spanning trees

1. Introduction

The present work addresses the terminal layout problem which consists of finding the best way to link \( n \) terminals, at different locations, to a central node (usually, the computer site). The optimal topology for this problem, corresponds to a tree in a graph \( G = (V,E) \) with all but one of the vertices in \( V \) corresponding to the terminals. The remaining node, the root node, refers to the computer site and edges in \( E \) correspond to the feasible wiring. The problem defined in this way corresponds to the well-known minimal spanning tree problem and can be solved efficiently (see, for instance, Prim, 1957). However, in the context of telecommunications networks, the terminal network involves certain constraints which are either related with the performance of the corresponding network or with the availability of some classes of devices. Three types of such constraints are: degree constraints which limit the number of edges incident to some of the nodes in the network (see Gavish, 1982 and Volgenant, 1989), capacity constraints which limit the number of nodes in each multipoint line (see Gavish, 1985 and Gouveia, 1995)
and hop constraints which limit the number of intermediate nodes between the root and any terminal (see Gouveia, 1996).

In this paper, we discuss the terminal layout problem with one additional constraint limiting the number of terminals with degree 1 in the tree solution. This type of constraint appears when designing tree-like networks and the software and hardware associated to each terminal differs accordingly with its position in the tree. Usually, the software and hardware associated to a "degree-1" terminal is cheaper than the software and hardware used in the remaining terminals because for any intermediate terminal \( j \) one needs to check if the arrival message is destined to that node or to any other node located after node \( j \). Therefore, that particular terminal needs software and hardware for message routing. On the other hand, such equipment is not needed in "degree-1" terminals. We also assume that the hardware and software for message routing in the nodes is already available, i.e., we know that \( p \) terminals may use such type of equipment. This means that the degree of the remaining terminals (let \( k \) denote the number of such terminals) have to be equal to 1 in the tree topology. This motivates a constraint stating that the tree solution has to contain at least \( k \) terminals with degree equal to 1. Notice that the solution may contain more than \( k \) terminals with degree 1 which means that the equipment associated to a few of these terminals is more expensive than it should be. To prevent such situations from happening, we look for spanning tree solutions with exactly \( k \) "degree-1" terminals. In Section 5 we describe generalizations of the proposed models which ignore the leaf constraint and consider layout costs together with equipment purchasing costs.

In the Graph Theory terminology the term "leaf" is used for denoting an edge which is incident to a node of degree 1. However, from the point of view of the terminal layout problem it is irrelevant whether the root node has, or not, degree equal to 1 in the tree solution. Therefore, in the context of this paper we exclude from the definition of "leaf", any edge incident to the root if that edge is the only edge incident to the root. With such an assumption, the above constraint can be restated as "the tree solution has to contain exactly \( k \) leaves" and the Leaf Constrained Minimal Spanning Tree (LMST) problem can be seen as the following graph theoretical problem:

Consider a graph \( G = (V, E) \) where \( V = \{0,1,\ldots,n\} \), with costs \( c_{ij} \) for each edge \((i,j) \in E\) and a natural number \( k \) \((1 \leq k \leq n)\). We want to find the minimal spanning tree such that the number of leaves is equal to \( k \).

The LMST is NP-hard because it contains as a particular case, the case \( k = 1 \), which is the shortest Hamiltonian path problem between a given node, node 0, and any other node. This problem is equivalent to the well-known traveling salesman problem and hence, the LMST is also NP-hard. A different combinatorial spanning tree problem which uses some of the concepts discussed in this paper is the "MaxLeaves" problem which consists of finding in a given graph, the spanning tree with the maximum number of leaves. Notice that this problem is trivial if the underlying graph is complete because any star solution is optimal. However, it is known to be NP-hard for general graphs (see Garey and Johnson, 1979). Several local search approximation algorithms for the MaxLeaves problem are described in Lu and Ravi (1992). More recently, Galbiati et al. (1994) have shown that no polynomial time approximation algorithm exists for this problem. As far as we know, no integer linear programming formulation for the MaxLeaves has been given in the literature. In Section 5 we shall show how to use the concepts discussed in this paper for obtaining two valid linear integer programming formulations for this problem.

This paper is organized in the following way. In Section 2, we present and discuss two directed formulations for the LMST. Two relaxation schemes for the LMST are also discussed in the same section. A local-exchange heuristic is discussed in Section 3. Computational results are reported in Section 4. In Section 5 we discuss formulations for related problems. Some conclusions are presented in Section 6.

2. Models for the LMST and lower bounding schemes

In this section we present two directed integer linear programming formulations for the LMST. In a
directed formulation, each edge \((i,j)\) \((i < j)\) is replaced by two arcs \((i,j)\) and \((j,i)\) with opposite directions. Additionally, the cost of the two arcs is equal to the cost of the original edge. Any edge adjacent to the root is replaced by only one single arc. The reason why we opted for modeling the problem in a directed graph is given by the fact that in this case it is much easier to model the leaf constraint. One of these formulations is directly derived from a reformulation, presented in Gouveia (1995), of a single-commodity flow based formulation for the directed minimal spanning tree (DMST) problem. The other formulation uses a DMST formulation as a starting formulation. Then, additional variables and additional constraints are added to the starting formulation in order to obtain a valid formulation for the LMST problem. We also describe two lower bounding schemes for the LMST. These schemes are derived from the two formulations discussed in this section.

2.1. A flow based formulation for the LMST

We start by presenting a single-commodity flow formulation for the DMST problem. This formulation is taken from Gavish (1982). Consider the binary variables \(X_{ij}\) such that \(X_{ij} = 1\) if arc \((i,j)\) is in the optimal solution and \(X_{ij} = 0\), otherwise. Consider also the non-negative variables \(Y_{ij}\) which represent the amount of flow produced by the root going through arc \((i,j)\). Finally, consider the vector \((d_0, d_1, \ldots, d_n)\) such that \(d_0 = 0\) and \(d_i = 1\) \((i = 1, \ldots, n)\). We have then the following formulation for the directed DMST:

Formulation FF (Gavish, 1982):

\[
\begin{align*}
\text{min} & \sum_{i=0}^{n} \sum_{j=1}^{n} c_{ij} X_{ij} \\
\text{subject to} & \\
\sum_{i=0}^{n} X_{ij} = 1, & j = 1, \ldots, n, \\
\sum_{i=0}^{n} Y_{ij} - \sum_{i=0}^{n} Y_{ji} = 1, & j = 1, \ldots, n, \\
X_{ij} \leq Y_{ij} \leq (n - d_i) X_{ij}, & i = 0, \ldots, n; \quad j = 1, \ldots, n, \\
X_{ij} \in \{0, 1\}, & i = 0, \ldots, n; \quad j = 1, \ldots, n.
\end{align*}
\]

To simplify the indexing, we have not considered variables \(X_{ii}\) and \(Y_{ii}\) \((i = 1, \ldots, n)\). Constraints (2) ensure that each node is in the solution. Constraints (3) are the flow conservation constraints for each terminal node. Constraints (3) and (4) guarantee that the solution does not contain circuits. It is easy to see that the flow in arc \((i,j)\) gives the number of nodes that are disconnected from the root if that arc is removed from the solution. A formal proof of the validity of this formulation for the DMST problem can be found in Gavish (1982).

Using the information associated to the variables involved in formulation FF we can easily see that a leaf corresponds to an arc \((i,j)\) with flow equal to one (i.e., the value of the corresponding \(Y_{ij}\) variable is equal to one). Therefore, a valid formulation for the LMST problem can be obtained by adding to FF a set of inequalities which state that

\[
\# \{(i,j) \mid i = 0, \ldots, n \text{ and } j = 1, \ldots, n : Y_{ij} = 1\} = k,
\]

where \#(S) denotes the number of elements in the set \(S\). However, it is far from easy to derive a set of linear inequalities which involve the two sets of variables \(\{X_{ij}, Y_{ij}\}\) and which are equivalent to Eq. (6). To overcome this problem we use a reformulation presented in Gouveia (1995) which is based on the fact that all the information provided by the \((X_{ij}, Y_{ij})\) variables can be duplicated by another set of variables. Let us consider the binary variables \(Z_{ijq}\) \((i = 0, \ldots, n; \quad j = 1, \ldots, n; \quad q = 1, \ldots, n - d_i)\) such that \(Z_{ijq} = 1\) if a flow of value \(q\) goes through arc \((i,j)\) and \(Z_{ijq} = 0\), otherwise. Notice that with this new set of variables, the condition (6) can be simply rewritten as one single inequality of the form:

\[
\sum_{i=0}^{n} \sum_{j=1}^{n} Z_{ij1} = k.
\]

The two following relations show how the original variables \(X_{ij}\) and \(Y_{ij}\) variables can be written in terms of the new variables.

\[
X_{ij} = \sum_{q=1}^{n-d_i} Z_{ijq}, \quad i = 0, \ldots, n; \quad j = 1, \ldots, n,
\]

\[
Y_{ij} = \sum_{q=1}^{n-d_j} qZ_{ijq}, \quad i = 0, \ldots, n; \quad j = 1, \ldots, n.
\]
For any arc \((i,j)\), relations (8) and (9) imply that \(X_{ij} = 1\) and \(Y_{ij} = p\) if and only if \(Z_{ijq} = 1\) and \(Z_{ijq} = 0\) for \(q = 1, \ldots, n - d_i\) and \(q \neq p\). Clearly, we also have \(X_{ij} = Y_{ij} = 0\) if and only if \(Z_{ijq} = 0\) for \(q = 1, \ldots, n - d_j\). If we replace in Eqs. (1)–(5) the \(X_{ij}\) and \(Y_{ij}\) variables by the corresponding right hand sides of Eq. (8) and Eq. (9) the following formulation is obtained:

Formulation FQ:

\[
\min \sum_{i=0}^{n} \sum_{j=1}^{n-d_i} \sum_{q=1}^{n-d_j} c_{ij} Z_{ijq} \tag{10}
\]

subject to

\[
\sum_{i=0}^{n} \sum_{q=1}^{n-d_i} Z_{ijq} = 1, \quad j = 1, \ldots, n, \tag{11}
\]

\[
\sum_{i=0}^{n} \sum_{q=1}^{n-d_i} qZ_{ijq} - \sum_{i=1}^{n} \sum_{q=1}^{n-1} qZ_{jiq} = 1, \quad j = 1, \ldots, n, \tag{12}
\]

\[
Z_{ijq} \in \{0, 1\}, \quad i = 0, \ldots, n; \quad j = 1, \ldots, n; \quad q = 1, \ldots, n - d_i. \tag{13}
\]

To simplify the indexing, we have not considered variables \(Z_{iiq}\) \((i = 1, \ldots, n)\). The main point of replacing in Eqs. (1)–(5) the \(X_{ij}\) and \(Y_{ij}\) variables by the corresponding right hand sides of Eq. (8) and Eq. (9) is that the set of constraints corresponding to the “copy” of constraints (4) in the new variables are always true and therefore they can be omitted from FQ. This leads to a formulation using only \(O(n)\) constraints and in fact, this was the main reason used in Gouveia (1995) to justify the use of this reformulation technique. Additionally, it was shown in the same work that the LP relaxations of the two formulations, FF and FQ, are equivalent, i.e., both formulations produce the optimal linear solutions with the same cost. Notice that using the original set of variables it seems to be rather difficult to write down the leaf condition (6). This work illustrates another reason for using this reformulation technique, namely that the third index can be easily used for writing down one single linear inequality which is equivalent to the leaf condition (6). In the sequel, LFQ refers to the formulation FQ “augmented” with the leaf constraint (7). The LFQ formulation involves a large number of variables \(O(n^3)\) but only a small number of constraints \((2n + 1)\). This number of constraints gives some indication that it would be worth trying to solve the corresponding LP relaxation for obtaining lower bounds on the optimal costs of LMST instances. Computational results given by solving this particular relaxation are discussed in Section 4. A few consequences of such results, also discussed in Section 4, indicate that the constraint (7) models the leaf requirement in a very weak way. These poor results motivate the formulation discussed in the next section.

2.2. An extended DMST formulation for the LMST

For deriving the next formulation we add a new set of variables and an adequate set of coupling constraints to a valid formulation for the DMST problem. This formulation uses only the binary arc design variables \(X_{ij}\), i.e., \(X_{ij} = 1\) if arc \((i,j)\) is included in the optimal solution and \(X_{ij} = 0\), otherwise. Then, the DMST can be formulated as follows:

Formulation X-DMST:

\[
\min \sum_{i=0}^{n} \sum_{j=1}^{n} c_{ij} X_{ij} \tag{14}
\]

subject to

\[
\sum_{i=0}^{n} X_{ij} = 1, \quad j = 1, \ldots, n, \tag{15}
\]

\[
\sum_{i \in S} X_{ij} \leq |S| - 1, \quad \forall S \subseteq \{1, \ldots, n\} : |S| \geq 2, \tag{16}
\]

\[
X_{ij} \in \{0, 1\}, \quad i = 0, \ldots, n; \quad j = 1, \ldots, n. \tag{17}
\]

To simplify the indexing, we have not considered variables \(X_{ii}\) \((i = 1, \ldots, n)\). Notice that the constraints (16) are the well-know subtour elimination constraints. One interesting feature of the above formulation is that it is exact in the sense that its optimal linear programming (LP) solution is always integer (see Edmonds, 1968). To formulate the LMST we add the binary variables \(U_j\), \(j = 1, \ldots, n\), such that \(U_j = 1\) if node \(j\) has degree 1 in the solution and \(U_j = 0\), otherwise. The constraint on the number of leaves can now be written as

\[
\sum_{j=1}^{n} U_j = k. \tag{18}
\]
Clearly, we need to add the following set of constraints which link the $X_{ij}$ variables with the $U_j$ variables

$$1 - U_i \leq \sum_{j=1}^{n} X_{ij} \leq (n - 1)(1 - U_i), \quad i = 1, \ldots, n. \quad (19)$$

These constraints guarantee that node $i$ has degree 1 ($U_i = 1$) if and only if no arcs are leaving that node. Notice also that the right-hand inequality of Eq. (19) can be tightened to

$$1 - U_i \leq \sum_{j=1}^{n} X_{ij} \leq k(1 - U_i), \quad i = 1, \ldots, n \quad (19')$$

because the outdegree of any node cannot be greater than $k$. Finally, the $\{0,1\}$ requirements for the new variables have to be included in the model:

$$U_j \in \{0,1\}, \quad j = 1, \ldots, n. \quad (20)$$

In the following, let XU-LMST denote the formulation defined by Eqs. (14)-(20). Notice also that from now on we use Eq. (19') instead of Eq. (19).

We could have used the FF formulation has a starting DMST formulation for this extended formulation. However, the LP relaxation of the FF formulation is weaker than the LP relaxation of the X-DMST formulation. It is also well known that in many cases, the LP relaxation of a given formulation gives an indication of the “strength” of the lower bounding methods which can be derived directly from such formulation. As we shall see later, with our choice we guarantee that the bound derived from the XU-LMST formulation is never worse than the trivial bound given by cost of the corresponding unconstrained minimal spanning tree. Before describing the proposed relaxation, we start by pointing out that the coupling constraints (19') can be splitted into the two following sets of constraints:

$$\sum_{j=1}^{n} X_{ij} \geq 1 - U_i, \quad i = 1, \ldots, n, \quad (19'a)$$

$$\sum_{j=1}^{n} X_{ij} \leq k(1 - U_i), \quad i = 1, \ldots, n. \quad (19'b)$$

To define the next Lagrangean relaxation we attach non-negative multipliers $\lambda_i$, $i = 1, \ldots, n$, to the constraints (19'a) and dualize them in the usual Lagrangean way (Geoffrion, 1974). After some manipulation of the objective function we obtain the following relaxed problem:

$$\min \left\{ \sum_{j=1}^{n} c_{0j} X_{0j} + \sum_{i=1}^{n} \sum_{j=1}^{n} \left( c_{ij} - \lambda_i - \mu_j \right) X_{ij} \right\}$$

$$- \sum_{i=1}^{n} \left( \lambda_i + k \mu_j \right) U_i + \sum_{i=1}^{n} \left( \lambda_i + k \mu_j \right)$$

subject to Eq. (15), Eq. (16), Eq. (17), Eq. (18) and Eq. (20).

Notice that for a given set of multipliers $\lambda_i$ and $\mu_j$ ($i = 1, \ldots, n$) the relaxed problem RLU $\triangle$ can be separated into two subproblems. The first one, REL$_{X_i}$, involves only the $X_{ij}$ variables and corresponds to a directed minimal spanning tree problem which can be solved in polynomial time (see Fischetti and Toth, 1993). The other subproblem, REL$_{U_i}$, involves the $U_i$ variables and can be simply solved by inspection on the value of the costs of the $U_i$ variables. The value of the $U_i$ variables corresponding to the $k$ smallest costs ($\lambda_i + k \mu_j$) is set to 1. The value of the remaining variables is set to 0. After solving both subproblems, the optimal value of RLU $\triangle$ is given by

$$v(\text{RLU} \triangle) = v(\text{REL}_{X_i}) + v(\text{REL}_{U_i})$$

$$+ \sum_{i=1}^{n} \left( \lambda_i + k \mu_j \right),$$

where $v(R)$ indicates the optimal value of problem $R$. An approximation of the optimal multipliers can be obtained by standard subgradient optimization (see Held et al., 1974). It can be easily seen that the relaxed problem satisfies the integrality property (see Geoffrion, 1974) and, therefore, the best lower bound that can be obtained by this procedure is, in theory, equal to the cost of the optimal solution of XU-LMST. However, considering the fact that an exponential number of constraints (constraints (16)) are involved in XU-LMST, the use of Lagrangean based methods seems to be a reasonable alternative to compute such a bound. Notice also that this bound is never worse than the bound given by computing the corresponding minimal spanning tree. This is a con-
sequence of the previously mentioned result due to Edmonds and to the fact that formulation XU-LMST is obtained by extending formulation X-DMST. In fact, we shall see in Section 4 that for a certain class of instances (instances with \( k \) small) this lower bounding scheme improves on lower bounds given by the costs of the corresponding minimal spanning trees.

3. A heuristic for the LMST

For obtaining upper bounds on the optimal values of LMST instances we used a 2-phase transformation heuristic. A spanning tree solution with at least \( k \) leaves is obtained in the first phase. If such a solution contains exactly \( k \) leaves it already satisfies the leaf constraint and we stop. Otherwise, we go to the second phase where a local-exchange procedure is used. In each iteration of this procedure, the current tree solution is transformed into another tree solution with a smaller number of leaves. As soon as we obtain a solution with \( k \) leaves, the local-exchange phase stops. The heuristic is summarized by the following pseudo-code:

\[
\text{Step 1. Obtain } \text{MST}(\geq k). \\
\text{Step 2. If the number of leaves in } \text{MST}(\geq k) \text{ is equal to } k \text{ STOP.}
\]

Fig. 1. (a) \( T_2 \) has one leaf less than \( T_1 \). (b) \( T_2 \) has one leaf less than \( T_1 \). (c) \( T_2 \) has one leaf less than \( T_1 \). (d) \( T_2 \) has two leaves less than \( T_1 \). (e) \( T_2 \) has one leaf less than \( T_1 \). (f) \( T_2 \) has one leaf less than \( T_1 \). (g) \( T_2 \) has two leaves less than \( T_1 \).
Otherwise, start a local-exchange procedure to decrease the number of leaves in the solution.

In the above description, MST(≥ k) denotes a spanning tree with at least k leaves. To obtain such a spanning tree we follow the following procedure:

1. select a subset of nodes \( K \subseteq V \setminus \{0\} \) such that \( |K| = k \) (the set \( K \) contains \( k \) nodes);
2. find the minimal tree spanning \( V \) such that each node in \( K \) has degree 1.

The minimal tree defined in (2) can be obtained in the following way: (a) obtain the minimal spanning tree in the subgraph induced by \( V \setminus K \); (b) for each node \( v \in K \), add the least cost arc linking any node in \( V \setminus K \) with node \( v \). The proof that such a procedure finds the least cost spanning tree defined in (2) follows from the fact that such a solution is obtained by combining together two least cost sets of arcs, one obtained in (a) and the other obtained in (b). To obtain the initial subset \( K \) we selected the \( K \) nodes (except the root node 0) which have smallest degree in the minimal tree spanning \( V \). Another rule which consists of selecting the \( K \) farthest nodes from the root was also described in Fernandes (1994). The quality of the heuristic solutions obtained with such a rule was much worse than the quality of the solutions obtained with the "\( k \) nodes with smallest degree" rule.

If the spanning tree obtained in step 2 contains exactly \( k \) leaves we stop. Otherwise, we start the local-exchange phase. One has to distinguish valid local-exchanges (a local exchange which reduces the number of leaves) from non-valid local exchanges (the number of leaves is not reduced). To describe such exchanges let \( e_1 \) denote the edge which is going to be removed from the current solution and let \( e_2 \) denote the edge to be included in such solution. For clarification, we use Figs. 1a–g to illustrate all of the exchanges we use in the second phase. In the figures, the root node is represented by a bold triangle while nodes of degree 1 are represented by empty squares. \( T_1 \) and \( T_2 \) are used to denote the two solutions, one \( (T_1) \) before the exchange and the other \( (T_2) \) after the exchange.

Valid-exchanges are discussed in Cases 1 to 7 presented next:

**Case 1.** One endpoint of \( e_1 \) is the root and has degree equal to 1. The degree of the other endpoint is greater or equal to 3. One of the endpoints of \( e_2 \) is the root and the other endpoint has degree equal to 1 in \( T_1 \) (Fig. 1a).

**Case 2.** One endpoint of \( e_1 \) is the root and has degree greater or equal to 2. The other endpoint of \( e_1 \) has degree less or equal to 2 and coincides with one of the endpoints of \( e_2 \). The degree of the other endpoint of \( e_2 \) in \( T_1 \) should be equal to 1 (Fig. 1b).

**Case 3.** One endpoint of \( e_1 \) is the root and has degree greater or equal to 2. The other endpoint of \( e_1 \) has degree equal to 2 and both endpoints of \( e_2 \) have degree 1 in \( T_1 \) (Fig. 1c).

**Case 4.** One endpoint of \( e_1 \) is the root and has degree greater or equal to 2. The other endpoint of \( e_1 \) has degree greater or equal than 3 and one of the endpoints of \( e_2 \) has degree equal to 1 in \( T_1 \) — in fact, if both endpoints of \( e_2 \) have degree 1 in \( T_1 \) then the local exchange reduces the number of leaves in two (Fig. 1d).

**Case 5.** No endpoint of \( e_1 \) is the root. One endpoint of \( e_1 \) has degree less or equal than 2 while the other has degree greater or equal to 3. One of the endpoints of \( e_2 \) coincides with the endpoint of \( e_1 \) which has degree less or equal than 2 while the other has degree 1 in \( T_1 \) (Fig. 1e).

**Case 6.** No endpoint of \( e_1 \) is the root. One endpoint of \( e_1 \) has degree 2 while the other has degree greater or equal to 3. Both endpoints of \( e_2 \) have degree equal to 1 in \( T_1 \) (Fig. 1f).

**Case 7.** No endpoint of \( e_1 \) is the root. Both endpoints of \( e_1 \) have degree greater or equal to 3. One of the endpoints of \( e_2 \) has degree 1 in \( T_1 \) — additionally, if both endpoints of \( e_2 \) have degree 1 in \( T_1 \) then the local exchange reduces the number of leaves in two (Fig. 1g).
Given a local exchange defined by two edges $e_1$ and $e_2$, let $S(e_1, e_2)$ denote the saving associated to that local exchange, i.e., $S(e_1, e_2) = \text{cost}(e_2) - \text{cost}(e_1)$. Our local-exchange procedure selects from every valid exchange the one that minimizes the savings function $S$. Notice that if at any given iteration, we have a solution with $k + 1$ leaves, the local-exchanges for the next iteration are selected from the ones which only reduce by one the current number of leaves. This heuristic has been embedded in the two Lagrangean schemes, i.e., in each iteration of the subgradient optimization procedure the minimum spanning tree solution is transformed into a feasible solution.

4. Computational results

In this section we compare the best lower bounds with the best upper bounds produced by the methods described in the previous two sections. The results comparing the different bounds are taken from a class of symmetric instances with up to 40 nodes. The cost matrix was taken as the integer part of the Euclidean distance between the coordinates of $(n + 1)$ points in a square grid of dimension 100 by 100. The points were randomly distributed in this grid. Several different values for $k$ were used for these tests. Two root locations were tested, one in the center of the grid and the other at the corner of the grid. For each combination of the parameter values, five problems were solved. Each method was implemented in FORTRAN and the tests were run on a PC 486/66Mhz. The first two columns indicate the parameters for each group of instances. Lower bounding information is given in the next six columns. A column with label "M1" indicates the average values of the ratio $[(v(\text{UB}) - v(M1))/v(\text{UB})]$ (where $v(M1)$ indicates the best lower bound given by method M1) taken over the five corresponding tests and UB gives the best upper bound obtained by the heuristic embedded in the Lagrangean relaxation. With respect to the Lagrangean based schemes we report two different lower bounds, one obtained after 100 iterations, the other obtained after 500 iterations. We have considered four possible cases for M1 which are specified below:

- MST = lower bound given by the cost of the corresponding minimal spanning tree.
- LFO = lower bound given by solving the LP relaxation of LFO (the lower bound is rounded to the next integer because, all instances have integer costs).
- Lag($k$) = lower bound given by the method described in the Section 2.2 after 100 or 500 iterations of the subgradient optimization procedure.
- Lag($n - 1$) = the same as above, but now we use the set of constraints (19) instead of the stronger set (19').

Notice that constraints (19) are obtained from Eq. (19') by replacing the parameter $k$ by $(n - 1)$. The reason for comparing Lag($k$) with Lag($n - 1$) is to access the effect of tightening constraints (19) into Eq. (19').

The last columns indicate the CPU times given in seconds by the Lagrangean schemes. The CPU times for the two schemes were independent of the version used, Lag($n - 1$) or Lag($k$), therefore we only report the CPU times given by the stronger Lag($k$).

For most of the instances tested, the bounds given by the Lagrangean schemes were not better than the MST bounds. Only for small cases of $k$ the Lagrangean schemes produced improvements on the MST bounds. However, for all of these cases the lower bound found by Lag($k$) was within 1% of the optimum. For these cases Lag($k$) produced reasonable improvements on the bounds given by Lag($n - 1$). For the remaining cases the difference was very small. To see the effect of increasing the number of iterations of the subgradient optimization procedure corresponding to the Lagrangean based schemes we compared the lower bounds obtained after two maximum limits, 100 iterations and after 500 iterations. Substantial improvements were obtained again for the instances with the smallest value of $k$ when the maximum number of iterations was increased to 500. For most of the instances with large values of $k$ we did not run the lower bounding schemes with up to 500 iterations because the results for the instances with the second smallest value of $k$ indicated that no improvements were expected. The performance of the proposed methods decreased considerably when the value of $k$ gets bigger. These results indicate that
for these instances, the LP relaxation of the XU-DMST model is very weak.

The results given in the column with label “LFQL” indicate that for every instance tested, the bounds given by the LP relaxation of LFQ, LFQL, were even worse (in many cases, much worse) than the trivial bounds given by costs of the corresponding minimal spanning trees (see column “MST”). Notice that the set of feasible solutions defined by LFQ is contained on the set of all directed minimal spanning trees. However, it is well known that the LP relaxation of FQ (which is a reformulation of FF) gives a very poor representation of the tree polytope and this might explain why the LFQ (which is FQ plus the leaf constraint) works very poorly in terms of the associated LP relaxation. For any instance tested the optimal value of LFQL was practically the same for different values of k. This might be explained by the fact that LFQL contains many feasible solutions with the same cost and also gives an indication of the weakness of the leaf constraint (7). To illustrate this situation consider the following two solutions (we need not to specify the associated edge costs because this situation happens frequently for several edge costs). The first solution (solution 1) corresponds to an optimal MST solution and contains two leaves (this solution is given in terms of the variables involved in the formulation LFQ). When we solved the LP relaxation of LFQ with k = 3 we expected to improve on the bound given by the MST solution. However, we obtained a different solution (solution 2) which has exactly the same cost as the optimal MST:

Solution 1.
\[ Z_{015} = Z_{122} = Z_{142} = Z_{231} = Z_{451} = 1; \]
other variables equal to zero.

Solution 2.
\[ Z_{015} = Z_{231} = Z_{451} = 1; \]
\[ Z_{121} = Z_{123} = Z_{141} = Z_{143} = 1/2; \]
other variables equal to zero.

The only effect of including the leaf constraint (7) in the LFQ formulation was to produce a splitting of the flow in arcs (1,2) and (1,4). In fact, the arc (1,2) which carries a flow equal to 2 in the MST solution was replaced by two “half-arcs”, one carrying a flow equal to 3 and the other carrying a flow equal to 1. Notice that the combined cost of these two half-arcs is equal to the cost of the original arc in the MST solution. A similar situation happens with respect to arc (1,4). This process of “flow-splitting” has already been pointed out in Gouveia (1995) to explain the weakness of some valid inequalities for the Capacitated Minimal Spanning Tree.

Table 1  
Best “gaps” for a set of complete graphs with the root in the center

<table>
<thead>
<tr>
<th>n</th>
<th>k</th>
<th>LFQL</th>
<th>MST</th>
<th>Lag(n-1)</th>
<th>Lag(k)</th>
<th>Lag(n-1)</th>
<th>Lag(k)</th>
<th>Lag(n-1)</th>
<th>Lag(k)</th>
<th>CPU times</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>4</td>
<td>0.095</td>
<td>0.010</td>
<td>0.007</td>
<td>0.003</td>
<td>0.007</td>
<td>0.001</td>
<td>8.2</td>
<td>33.4</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>0.095</td>
<td>0.009</td>
<td>0.009</td>
<td>0.009</td>
<td>0.009</td>
<td>0.009</td>
<td>5.3</td>
<td>26.1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>12</td>
<td>0.189</td>
<td>0.089</td>
<td>0.089</td>
<td>0.089</td>
<td>0.089</td>
<td>0.089</td>
<td>19.3</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>16</td>
<td>0.307</td>
<td>0.236</td>
<td>0.236</td>
<td>0.236</td>
<td>0.236</td>
<td>0.236</td>
<td>28.0</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>5</td>
<td>0.094</td>
<td>0.017</td>
<td>0.015</td>
<td>0.012</td>
<td>0.015</td>
<td>0.006</td>
<td>128.9</td>
<td>643.6</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>10</td>
<td>0.078</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>15</td>
<td>0.099</td>
<td>0.022</td>
<td>0.022</td>
<td>0.022</td>
<td>0.022</td>
<td>0.022</td>
<td>156.5</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>0.142</td>
<td>0.069</td>
<td>0.069</td>
<td>0.069</td>
<td>0.069</td>
<td>0.069</td>
<td>273.8</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>25</td>
<td>0.207</td>
<td>0.139</td>
<td>0.139</td>
<td>0.139</td>
<td>0.139</td>
<td>0.139</td>
<td>360.0</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>30</td>
<td>0.295</td>
<td>0.236</td>
<td>0.236</td>
<td>0.236</td>
<td>0.236</td>
<td>0.236</td>
<td>432.3</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>35</td>
<td>0.424</td>
<td>0.382</td>
<td>0.382</td>
<td>0.382</td>
<td>0.382</td>
<td>0.382</td>
<td>486.9</td>
<td>a</td>
<td></td>
</tr>
</tbody>
</table>

Note: “a” indicates that the corresponding tests were not run.
Table 2
Best "gaps" for a set of complete graphs with the root in the corner

<table>
<thead>
<tr>
<th>n</th>
<th>k</th>
<th>100 iterations</th>
<th>500 iterations</th>
<th>CPU times</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>LFQ</td>
<td>MST</td>
<td>Lag(n - 1)</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>0.124</td>
<td>0.010</td>
<td>0.009</td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>0.089</td>
<td>0.013</td>
<td>0.013</td>
</tr>
<tr>
<td>20</td>
<td>12</td>
<td>0.177</td>
<td>0.095</td>
<td>0.095</td>
</tr>
<tr>
<td>20</td>
<td>16</td>
<td>0.294</td>
<td>0.265</td>
<td>0.265</td>
</tr>
<tr>
<td>40</td>
<td>5</td>
<td>0.072</td>
<td>0.013</td>
<td>0.012</td>
</tr>
<tr>
<td>40</td>
<td>10</td>
<td>0.060</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>40</td>
<td>15</td>
<td>0.085</td>
<td>0.028</td>
<td>0.028</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>0.129</td>
<td>0.074</td>
<td>0.074</td>
</tr>
<tr>
<td>40</td>
<td>25</td>
<td>0.197</td>
<td>0.146</td>
<td>0.146</td>
</tr>
<tr>
<td>40</td>
<td>30</td>
<td>0.287</td>
<td>0.242</td>
<td>0.242</td>
</tr>
<tr>
<td>40</td>
<td>35</td>
<td>0.416</td>
<td>0.392</td>
<td>0.392</td>
</tr>
</tbody>
</table>

Note: "a" indicates that the corresponding tests were not run.

The CPU times reported in Tables 1 and 2 indicate that in general the CPU times for the Lagrangean schemes increase when the value of k also increases. This is explained by the heuristic and the solution procedure for the second subproblem in the relaxation. We should point out that only 2.3 seconds of CPU were needed in the PC 486/66Mhz for obtaining the lower bounds given by Lag(n - 1) and Lag(k) after 100 iterations for the n = 40 tests and k = 5 if the heuristic is removed from the Lagrangean scheme. This indicates that the embedded heuristic is the main contributor to the global CPU times reported in Tables 1 and 2. To obtain the LP bounds given by LFQ, we used the LPS-867 package for solving the associated LP model. In average, 4 min and 30 sec were needed for solving each LP model with n = 40 in the PC 486/66Mhz. On the other hand, only 5 sec were needed for solving the LP models associated to the n = 20 instances. The discrepancy in the CPU times for the two sets of tests can be explained by the number of variables (O(n^3)) involved in the LFQ model.

Table 3 shows what is gained by embedding the heuristic in the Lagrangean scheme. The entries indicate the average values of the ratio [(v(OUB) - v(FUB))/v(OUB)] (where v(OUB) indicates the original upper bound and v(FUB) indicates the final upper bound reported by the Lagrangean scheme) taken over the five corresponding tests. These results indicate that the best improvements are obtained precisely for the most difficult tests, namely the ones with big values of k.

5. Related models

As pointed out in the introduction we may ignore the leaf constraint and combine the layout costs with the equipment purchasing costs. To do that, let D denote that cost of purchasing equipment needed for message routing for one single node. Then, the ob-
Objective function of the FQ model would be modified to

\[
\min \sum_{i=0}^{n} \sum_{j=1}^{n} \sum_{q=1}^{n} c_{ijp} Z_{ijq},
\]

where

\[
c_{ijp} = c_{ij}, \quad i = 0, \ldots, n; \quad j = 1, \ldots, n; \quad p = 1, \]

\[
c_{ijp} = D + c_{ij}, \quad i = 0, \ldots, n; \quad j = 1, \ldots, n; \quad p = 2, \ldots, Q - d_i.
\]

Alternatively, we may use the model XU-LMST and modify the objective function to

\[
\min \sum_{i=0}^{n} \sum_{j=1}^{n} c_{ij} X_{ij} + \sum_{j=1}^{n} D(1 - U_j).
\]

The other related problem is the MaxLeaves problem also mentioned in the introduction. Notice that for formulating this problem we also need to identify when the root node is an endpoint of a leaf. The FQ model is a network flow model where each terminal node receives one unit of flow and the root node send \(n\) units of flow. The root node is an endpoint of a leaf if all the flow sent by the root (\(n\) units of flow) flows along a single arc leaving the root. Then, this arc is also a leaf. Using the variables involved in the FQ formulation this means that for one single \(j\) \((j = 1, \ldots, n)\) the corresponding variable \(Z_{0jn}\) is equal to 1. To formulate the MaxLeaves problem one only needs to add the following objective function to Eqs. (11)-(13):

\[
\max \sum_{j=1}^{n} Z_{0jn} + \sum_{i=0}^{n} \sum_{j=1}^{n} Z_{ij1}.
\]

With respect to the XU-LMST model, we need to introduce a new binary variable, \(U_0\), which indicates whether node 0 is an endpoint of a leaf. Clearly, we need to add the following set of constraints which link the \(X_{0j}\) variables with the \(U_0\) variable:

\[
2 - U_0 \leq \sum_{j=1}^{n} X_{0j} \leq (n - 1)(1 - U_0) + 1.
\]

\(i = 1, \ldots, n.\)

The above constraints guarantee that the root node 0 has outdegree 1 \((U_0 = 1)\) if and only if one single arc is leaving that node. If we add such constraints to Eqs. (15)-(19) and use the following objective function

\[
\max \sum_{j=0}^{n} U_j
\]

we obtain a valid formulation for the MaxLeaves problem.

6. Conclusion

In this paper we have described formulations, lower bounding schemes and upper bounding schemes for a minimal spanning tree problem with a constraint on the number of leaves. We have motivated the inclusion of this type of constraints with a real world problem which appears in the design of terminal networks. Our upper bounding and lower bounding schemes perform rather well for the cases where the required number of leaves is small. However, the reported gaps are substantially large for the other cases. Moreover, the results also indicate that for these cases our best lower bounds were not better than the trivial lower bounds given by the cost of the corresponding unconstrained minimal spanning tree problem. This indicates that for such cases, we have to search for new inequalities which cut off many of the optimal solutions of the LP relaxation of the XU-DMST model. It is also interesting to point out that the two solutions given in Section 5 involving the \(Z_{ijq}\) variables are transformed, by Eq. (8), into the same solution involving the \(X_{ij}\) variables. This solution is defined by \(X_{01} = X_{12} = X_{14} = X_{23} = X_{45} = 1\) (other variables equal to zero). By choosing \(U_1 = 0, U_2 = U_4 = 1/2\) and \(U_3 = U_5 = 1\), we see that this combined \(\{X, U\}\) solution is feasible for the LP relaxation of XU-LMST when \(k = 3\). This deceptively simple solution indicates that one area of research is to tighten or generalize the coupling constraints (19').

Acknowledgements

The authors thank Prof. F. Maffioli for bringing the MaxLeaves problem to our attention. Thanks are also to two referees whose suggestions have also helped to improve the paper.
References


