J Heuristics (2009) 15: 133–151
DOI 10.1007/s10732-008-9079-x

GRASP with hybrid heuristic-subproblem optimization for the multi-level capacitated minimum spanning tree problem

Alexandre X. Martins · Mauricio C. de Souza · Marcone J.F. Souza · Túlio A.M. Toffolo

Received: 16 February 2007 / Revised: 6 February 2008 / Accepted: 24 April 2008 /
Published online: 8 May 2008

Abstract We propose a GRASP using an hybrid heuristic-subproblem optimization approach for the Multi-Level Capacitated Minimum Spanning Tree (MLCMST) problem. The motivation behind such approach is that to evaluate moves rearranging the configuration of a subset of nodes may require to solve a smaller-sized MLCMST instance. We thus use heuristic rules to define, in both the construction and the local search phases, subproblems which are in turn solved exactly by employing an integer programming model. We report numerical results obtained on benchmark instances from the literature, showing the approach to be competitive in terms of solution quality. The proposed GRASP have in fact improved the best known upper bounds for almost all of the considered instances.

Keywords Multi-level capacitated minimum spanning tree problem · Network design · GRASP · Subproblem optimization · Local search

A.X. Martins
Departamento de Ciências Exatas e Aplicadas, Universidade Federal de Ouro Preto, João Monlevade, MG, Brazil
e-mail: xmartins@decea.ufop.br

M.C. de Souza (✉)
Departamento de Engenharia de Produção, Universidade Federal de Minas Gerais, Caixa Postal 4006, cep: 31250-970, Belo Horizonte, MG, Brazil
e-mail: mauricio.souza@pq.cnpq.br

M.J.F. Souza
Departamento de Computação, Universidade Federal de Ouro Preto, Ouro Preto, MG, Brazil
e-mail: marcone@iceb.ufop.br

T.A.M. Toffolo
Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil
e-mail: tuliotoffolo@yahoo.com.br

1 Introduction

Capacitated trees arise in the context of centralized network design problems when constraints limit the traffic transmission supported by a link. These problems consist of finding a minimal cost spanning tree rooted at a central node such that the amount of traffic to be transferred from the central to the other nodes is bounded by edge capacities. Applications can be found, for instance, in communication networks, e.g., Esau and Williams (1966), Gavish (1982, 1991), Rothfarb and Goldstein (1971).

Esau and Williams (1966) were the first, to our knowledge, to introduce the Capacitated Minimum Spanning Tree (CMST) problem. Let $G = (V, E)$ be a connected undirected graph, where V is the set of nodes and E is the set of edges. Non-negative costs c_{ij} and weights b_i are associated respectively to each edge $(i, j) \in E$ and to each node $i \in V$. Given an integer Q and a central node $r \in V$, the CMST problem consists of finding a minimum spanning tree T of G such that the sum of the node weights in each connected component of the subgraph induced in T by $V - \{r\}$ is less than or equal to Q. The CMST problem is NP-Hard for $3 \leq Q \leq \frac{|V|}{2}$, Papadimitriou (1978). Both exact and heuristic methods have been developed for the CMST problem, see for instance Ahuja et al. (2001), Amberg et al. (1996), Gavish (1982), Gouveia (1995), Gouveia and Martins (2000, 2005), Hall (1996), Martins (2007), Sharaiha et al. (1997), Souza et al. (2003) and Uchoa et al. (2008).

In the CMST problem, a capacity Q is installed on every edge (i, j) composing a feasible tree T, paying thus its full cost c_{ij} no matter the traffic to be transferred between nodes i and j. A natural extension to the CMST problem is to consider that between each pair of nodes a feasible set of capacities with different costs is available. This means that instead of installing a capacity Q on every edge of T, choice can be made among different values of capacities and respective costs. This problem has been recently treated by Gamvros et al. (2003, 2006) as the Multi-Level Capacitated Minimum Spanning Tree (MLCMST) problem.

Other problems having characteristics in common with the MLCMST problem have been treated in the literature. Rothfarb and Goldstein (1971) have treated in the early seventies a similar problem, denoted one-terminal telpak problem. That work concerns a network design problem in which requirements from locations to a common facility must be satisfied minimizing a nonlinear neither convex nor concave but piecewise linear cost function. Network design problems related to the MLCMST can be found in telecommunications, see Berger et al. (2000) and Gavish (1991), and in gas and oil pipeline systems, see Brimberg et al. (2003).

In this paper we propose a GRASP for the MLCMST problem that exploits an hybrid heuristic-subproblem optimization scheme. Let us consider a collection $\Gamma = \{S_i\}$ of nonempty sets forming a partition of $V - \{r\}$ such that $\sum_{u \in S_i} b_u \leq Q$ for each S_i belonging to Γ, and let G_i be the subgraph of G induced by $S_i \cup \{r\}$. The motivation to employ subproblem optimization is that while in the CMST problem an instance defined by G_i reduces to the MST problem, this is not the case for the MLCMST problem. Indeed, for the MLCMST problem, to find a minimum cost capacitated spanning tree of a subgraph induced in G by $S_i \cup \{r\}$ may be itself a MLCMST instance. Gamvros et al. (2002, 2006), in their genetic algorithm, exploited this fact by partitioning $V - \{r\}$ and then trying to locally optimize each subproblem. We exploit this same search structure, but in our GRASP, heuristics are used to construct and rearrange a partition of $V - \{r\}$ generating smaller-sized MLCMST subproblems which are independently solved to optimality.

The paper is organized as follows. In the next section we define the MLCMST problem and present recent advances and an integer programming formulation as well. In Sects. 3 and 4, respectively, we describe the proposed GRASP and report computational experiments on benchmark instances. We make concluding remarks in the last section.

2 Problem definition

The MLCMST problem can be stated as follows. Let $G = (V, E)$ be a connected undirected graph, where V denotes the set of nodes and E denotes the set of edges. Let us consider L different capacities of value z^l, $l = 1, \dots, L$, such that $0 < z^1 < z^2 < \cdots < z^L$, which are available to be installed on each edge $(i, j) \in E$ with a cost c^l_{ij}. We consider that only one capacity can be installed on an edge. Given a spanning tree $T = (V, \hat{E})$ of G, $z^{\hat{l}}_{(i,j)}$ denotes the capacity installed on edge $(i, j) \in \hat{E}$. The cost of T is given by $\sum_{(i,j)\in\hat{E}} c^{\hat{l}}_{ij}$. A non-negative weight b_i is associated to each node $i \in V$. The tree T being rooted at a central node $r \in V$, the predecessor $p(i)$ of a node $i \in V - \{r\}$ is the first node in the path from i to r in T. We denote by T_i the connected component containing node i in the forest obtained by removing edge $(p(i), i)$ of T. The MLCMST problem consists of finding a minimum cost spanning tree T of G such that the sum of the node weights in each T_i, $i \in V - \{r\}$, is less than or equal to the capacity $z^{\hat{l}}_{(p(i),i)}$ installed on edge $(p(i), i)$. This problem clearly reduces to the CMST if only one capacity of value Q is available.

2.1 Recent advances

The MLCMST have been treated considering a cost structure that reflects the economies of scale effect, i.e., $\frac{c^l_{ij}}{z^l} > \frac{c^{l+1}_{i,j}}{z^{l+1}}$, for $l = 1, \dots, L-1$, and for all $(i, j) \in E$. Gamvros et al. (2002) proposed a mixed-integer programming model for the MLCMST problem in which each node $i \in V - \{r\}$ is the source of a commodity with unitary demand, and the central node is the destination of all the $|V| - 1$ commodities. They also developed a genetic algorithm that split the MLCMST into two separate subproblems: a grouping problem and a network design problem. The grouping problem consists of assigning nodes to components to be connected to the central node such that the component's node weights sum does not exceed the highest capacity z^L. The network design problem is in the worst-case a MLCMST instance in the subgraph induced by the nodes of a group together with the central node. As proposed by Falkenauer (1996), the chromosome represents a solution by breaking it up into two parts: a group part and an item part which are associated with the grouping and network design subproblems respectively. A crossover operator similar to that of Falkenauer (1996) and a mutation operator are applied.

Martins et al. (2005) conducted computational experiments on solving with a commercial package the MLCMST problem. In that work, three integer programming models for MLCMST were compared on instances adapted from benchmark ones introduced by Gouveia (1995) for the CMST. Two models are mixed integer flow formulations: the single commodity formulation due to Gavish (1982), and the multicommodity formulation proposed by Gamvros et al. (2002). The remaining model is a pure integer programming formulation: the $2n$ constraint formulation due to Gouveia (1995), i.e., $|V| = n$. The models due to Gavish (1982) and to Gouveia (1995) were originally proposed to the CMST problem, and they are straightforwardly adapted to the MLCMST. The $2n$ constraint model, in the CMST context, uses binary variables that limit to l, $l = 1, \dots, Q - b_i$ (b_r is assumed to be 0), the flow on edge (i, j). This leads to what we call a capacity-indexed model for the MLCMST problem. The capacity-indexed model has shown to be the most effective for the purpose of solving MLCMST instances to optimality, though the multicommodity flow model proposed by Gamvros et al. (2002) seems to provide tighter linear relaxation lower bounds. The capacity-indexed model was the only one able to solve instances up to 30 nodes in less than one hour (quite often in few seconds) and obtaining less than 10% of optimality gap for instances up to 40 nodes, running LINGO version 7.0 on a Pentium IV 1,8 GHz with 256 MB of RAM memory (Martins et al. 2005).

Gamvros et al. (2006) performed recently a comprehensive study on the MLCMST problem. The authors compared in terms of lower bounds two flow-based mixed integer programming formulations: a strengthened formulation of the single commodity flow model by Gavish (1982), denoted by ESCF, and a strengthened formulation of the multicommodity flow model (Gamvros et al. 2002), denoted by MCF. Computational experiments with medium-sized instances (up to 30 nodes) showed that for bounding purposes ESCF is most useful than MCF. The linear relaxation of ESCF is rapidly solved, while the relaxation of MCF generates more difficult linear programs providing only slightly better bounds.

Gamvros et al. (2006) also developed heuristic procedures. A saving construction heuristic starts with a star network in which each node $i \in V - \{r\}$ is connected to r with capacity $z^1_{(i,r)}$. Then, based upon savings in joining nodes in a same component, some edge capacities are upgraded from z^1 to z^L. When no more savings with capacity z^L are possible, the procedure searches for savings in upgrading from z^1 to z^{L-1}, and so on. As observed in the CMST problem with the classical Esau and Williams heuristic (Esau and Williams 1966), the proposed saving heuristic obtain good upper bounds for the MLCMST in reduced computational times. Gamvros et al. (2006) exploited the node-based, multi-exchange neighborhood structures developed by Ahuja et al. (2001) in two local search procedures for MLCMST problem. The cyclic and path exchanges define neighborhoods of exponential size which are explored by constructing an improvement graph. In the first variant, local search is directly applied to a solution obtained with the saving construction heuristic. In the second variant, called Randomized Start Local Search, different starting solutions are obtained by running the saving construction heuristic on perturbed cost versions of the original graph. Local search is then independently applied to each starting solution. The authors improved the genetic algorithm proposed in Gamvros et al. (2002) by applying a mutation operator based upon the cyclic and path exchange neighborhoods. Computational experiments were conducted on graphs with up to 150 nodes.

Three capacities with costs proportional to the Euclidean distances by factors reflecting economies of scale were considered. Numerical results obtained with the saving heuristic, the two local search procedures, and the genetic algorithm showed average gaps from lower bounds ranging from 6.09% to 9.91%, see Gamvros et al. (2006).

2.2 Capacity-indexed model

We now present the capacity-indexed formulation for the MLCMST problem. This model was originally proposed by Gouveia (1995), and has recently been used with success by Uchoa et al. (2008) in a branch-and-cut-and-price algorithm for the CMST. Based on our previous computational experience reported in Martins et al. (2005), we have chosen this model to solve to optimality the subproblems generated by the proposed GRASP.

The capacity-indexed model requires the following assumptions: (i) $z^1 = 1$ and (ii) capacities increase from 1 to z^L by unitary increments. This however may not be found in ordinary practice. In case conditions (i) and (ii) do not hold, we create artificial capacities $\bar{z}$. The number of different capacities available is set to $P = z^L$, and then $\bar{z}^1 = 1, \bar{z}^2 = 2, \dots, \bar{z}^P = z^L$. Given an edge (i, j), the cost $\bar{c}_{ij}^p$ associated to each artificial capacity $\bar{z}^p$, $p = 1, \dots, P$, is then set to

$$\bar{c}_{ij}^p = \begin{cases} c_{ij}^1, & p = 1, \dots, z^1, \\ c_{ij}^l, & p = z^{l-1} + 1, \dots, z^l,\ l = 2, \dots, L. \end{cases}$$

Although the MLCMST problem is defined on an undirected graph, we describe a model that works on a directed graph generated by replacing an edge (i, j) in the original graph by two arcs (i, j) and (j, i) keeping the same cost structure, e.g., (Gamvros et al. 2006; Martins et al. 2005; Uchoa et al. 2008). A binary variable x_{ij}^l, $i \in V$, $j \in V - \{r\}$, and $l = 1, \dots, L$ is defined such that

$$x_{ij}^l = \begin{cases} 1, & \text{if capacity } l \text{ is installed on arc } (i, j), \\ 0, & \text{otherwise.} \end{cases}$$

The capacity-indexed formulation is then written as:

$$\min \quad \sum_{p=1}^{P} \sum_{i \in V} \sum_{j \in V - \{r\}} \bar{c}_{ij}^p x_{ij}^p \tag{1}$$

$$\text{s.t.} \quad \sum_{p=1}^{P} \sum_{i \in V} x_{ij}^p = 1 \quad \forall j \in V - \{r\}, \tag{2}$$

$$\sum_{p=1}^{P} \sum_{i \in V} \bar{z}_{ij}^p x_{ij}^p - \sum_{p=1}^{P} \sum_{i \in V - \{r\}} \bar{z}_{ji}^p x_{ji}^p = b_i \quad \forall j \in V - \{r\}, \tag{3}$$

$$x_{ij}^p \in \{0, 1\} \quad \forall i \in V, \forall j \in V - \{r\}, p = 1, \dots, P. \tag{4}$$

Constraints Eq. 2 and Eq. 3 are respectively the in-degree constraint for an arborescence rooted at r and the capacity-balance constraint. These constraints together guarantee an arborescence feasible to the MLCMST problem, since Eqs. 2 and 3 prevent both cycles and sub-arborescences rooted at a node $i \in V - \{r\}$ with node weights exceeding arc $(p(i), i)$ capacity.

3 GRASP with hybrid heuristic-subproblem optimization

The proposed GRASP employs an hybrid heuristic-subproblem optimization scheme. Heuristic rules are used both to do a partition of $V - \{r\}$ and to decide whether to call an exact method within the construction and the local search phases. GRASP is a multi-start metaheuristic proposed by Feo and Resende (1995) which has been widely used to obtain good quality solutions for many combinatorial problems. The reader is referred to Festa and Resende (2001, 2004) and Resende and Ribeiro (2003) for in-depth surveys covering GRASP from basic scheme to recent enhancements, implementation strategies and successful applications. A GRASP iteration consists basically of two subsequent phases: construction phase and local search phase. The construction phase builds a feasible solution. The local search starts off with the solution built in the former phase and tries, by investigating neighborhoods, to achieve improvements until a local minima. The procedure returns the best solution found after `Max_It` iterations.

3.1 Construction phase

The construction phase builds in two steps a feasible solution to the MLCMST problem. We first use a greedy randomized heuristic to do a partition of $V - \{r\}$ in R_k, $k = 1, \ldots, K$ subsets. This step has an input parameter $w \geq z^L$ that limits the cardinality of each subset in the partition, thus $K = \lceil \frac{|V - \{r\}|}{w} \rceil$ in the unitary node weights case. We build one subset of the partition at a time. Let S denote the candidate nodes to be inserted in subset R_k being built, initially we set $S = V - \{r\}$ and $k = 1$. The procedure starts an iteration by removing a node i at random from S and inserting it in R_k. While $|R_k| < w$ and $S \neq \emptyset$ the procedure moves one node after the other from S to R_k. Thus, all nodes remaining in S are candidates to be inserted in R_k. We create then a restricted candidate list (RCL) formed by the best elements given by a greedy evaluation function. The RCL is formed by those nodes whose incorporation to R_k results in the smallest incremental cost according to Prim's algorithm to compute a MST, see for instance (Ahuja et al. 1993). Let d_j, for a node $j \in S$, be a label defined as $d_j = \min\{c^1_{ij} : i \in R_k\}$. We then set d_{min} and d_{max} respectively to the minimum and maximum values of d_j, $j \in S$. Given a parameter $\alpha \in [0, 1]$, RCL is defined as

$$\text{RCL} = \{j \in S : d_j \leq d_{min} + \alpha(d_{max} - d_{min})\}.$$

The element to be moved from S to R_k is randomly selected from those in the RCL, and labels d_j are updated for the nodes remaining in S following Prim's algorithmic logic (Ahuja et al. 1993). When w nodes are inserted in R_k, we increment k and proceed until a partition of $V - \{r\}$ is done.

At this point, there are K subproblems consisting each of an independent MLCMST instance on the subgraph induced in G by $R_k \cup \{r\}$, $k = 1, \ldots, K$. In the second step, we solve independently each of the K subproblems to optimality, and we have thus a feasible solution to the original MLCMST instance. For this purpose we employ the capacity-indexed model, cf., Sect. 2.2, by making calls to an optimization package. Note that by solving exactly the subproblems, nodes grouped in one subset R_k of the partition can actually form two or more components connected to r.

3.2 Local search phase

The local search phase tries to improve a feasible solution by re-arranging nodes of different components connected to r. Let us consider, in the sequel of this section, a spanning tree $T = (V, \hat{E})$ feasible to the MLCMST problem. Let us also designate a connected component of the forest obtained by the elimination of r and its incident edges from T as a component of T. A move is then defined by combining two or more components. The value of applying a move to T in order to generate a neighbor $T' = (V, \hat{E}')$ is given by $\Delta = \sum_{(i,j)\in \hat{E}'} c_{ij}^{\hat{l}'} - \sum_{(i,j)\in \hat{E}} c_{ij}^{\hat{l}}$, where $c_{ij}^{\hat{l}}$ denotes the cost of capacity $z_{(i,j)}^{\hat{l}}$ installed on edge $(i, j) \in \hat{E}$, cf., Sect. 2. The value of Δ is computed by obtaining the best way to connect to r the nodes of the components involved. As mentioned before, evaluate this kind of move means solving a smaller-sized MLCMST instance in the worst-case. Thus, we limit the size of the subproblem induced in G by the components defining a move. We make within the local search calls to an optimization package to solve exactly each MLCMST instance modeled with the capacity-indexed formulation. We propose thus heuristic rules to form the subproblems.

The rationale of the heuristic is to identify a local gain in connecting two nodes that are in different components of T. Given a node i, let us define by V_i the set of nodes in the component containing i, and let us also denote by $c(T_i)$ the cost of the subgraph induced in T by $V_i \cup \{r\}$ with the respectively installed capacities. Figure 1 shows in details the local search procedure. We consider the set S of non-leaf nodes in T as candidates to be "reference" nodes to form a subproblem. At each iteration of the local search, a node i is chosen at random from S. The procedure then starts to build a subset of nodes P which may induce a subproblem in G. Initially P is set to $V_i \cup \{r\}$, and the gain γ is set to $c(T_i)$. The set $\bar{S}$ contains the nodes that connect the other components to r. The parameter h, which limits the cardinality of P, is chosen at random from the interval $[\underline{h}, \bar{h}]$, where $\underline{h}$ and $\bar{h}$ are positive integers. We try then to add the nodes of one or more components to P so as to build a subproblem. A node j is chosen at random from $\bar{S}$, and the procedure looks for a node u belonging to V_j that could be connected to i with the capacity $\hat{l}$ installed on edge $(p(u), u)$ at a smaller cost. The component V_j is included in the subproblem being built if both there exists such a node and the cardinality of $V_j \cup P$ does not exceed h. In this case the gain is increased by adding $c(T_j)$ to γ. Node j is removed from $\bar{S}$ and the procedure continues trying to enlarge the subproblem until either the cardinality of P is h or $\bar{S}$ is empty.

At this point, a subproblem has been built if P contains also nodes other than the ones in $V_i \cup \{r\}$. We then solve to optimality the subproblem induced in G by P. That is, we try to re-arrange in an optimal manner the components of T selected to be included in P. Let us suppose that the optimal solution of the subproblem has a value of ϕ, note that ϕ cannot be greater than γ. If ϕ is smaller than γ, an improving move with value $\Delta = \phi - \gamma$ has been found, and the current solution T is updated. The set S of candidates to be "reference" nodes is also reconfigured. We include in (resp. remove from) S the non-leaf (resp. leaf) nodes of the optimal solution of the subproblem that are not (reps. are) already in it. The choice of a node i from S at the beginning of an iteration may not lead to an improving move for two reasons: the move value Δ is zero or none of the components has been added to P. In the first case the optimal solution of the subproblem has the same configuration the components involved already had in T, and we remove from S all the nodes of component V_i. In the second case the exact method has not been called, and we remove from S only the node i. The procedure iterates while S in not empty.

4 Computational results

In order to measure the potential of the proposed GRASP in finding good quality solutions, we conducted computational experiments on benchmark instances for the MLCMST problem. We report in this section numerical results on instances introduced in the literature by Gamvros et al. (2006). The authors introduced small instances with 20 and 30 nodes plus the central node, and larger ones with 50, 100 and 150 nodes plus the central node. These are unitary demand instances in which the nodes, except the central, are randomly distributed in a 40×40 square grid. There are three problem types for each size, according to the location of the central node: in the center, at the edge, and randomly located. For each problem type and size, 50 instances were generated. In all instances, three different capacities are available and their values are the same for every edge: $z^1 = 1$, $z^2 = 3$, and $z^3 = 10$. For each edge (i, j), the cost c^1_{ij} of the first facility is equal to Euclidean distance between their extremities (not rounded), and the cost of the second and third facilities are respectively $c^2_{ij} = 2c^1_{ij}$ and $c^3_{ij} = 3c^1_{ij}$.

Regarding the larger instances, Gamvros et al. (2006) have reported, for each combination number of nodes—location of the central node, aggregated percentage gaps (average, min and max) with respect to the lower bounds obtained by the LP relaxation of ESCF. For the 50 and 100 nodes instances, the authors have reported results for all the methods developed by them: the construction heuristic, the two local search procedures, and the genetic algorithm (Gamvros et al. 2006), cf., Sect. 2.1. The genetic algorithm seems to perform better, since it obtained the smallest average gaps for all the six problem type size pairs of the instances of 50 and 100 nodes. For the 150 nodes instances, Gamvros et al. (2006) have reported results for the construction heuristic and the first variant of the local search, cf., Sect. 2.1.

We considered in this study the three problem type instances with 50 nodes, and the central node located in the center instances with 100 and 150 nodes, resulting in a total of 250 instances. This option is due to the large computational times needed

procedure Local_Search
Set S as the set of non-leaf nodes in T;
while $(S \neq \emptyset)$ **do**
Select at random a node i from S.
$\gamma \leftarrow c(T_i)$
$P \leftarrow V_i \cup \{r\}$
$\bar{S} \leftarrow \{j \in V - P \mid (j, r) \in \hat{E}\}$
Chose at random $h \in [\underline{h}, \bar{h}]$
while $(|P| < h)$ and $(\bar{S} \neq \emptyset)$ **do**
Select at random a node j from $\bar{S}$.
For each node $u \in V_j$ let $\hat{l}$ be the capacity installed on edge $(p(u), u)$
if (there exists a node $u \in V_j$ such that $c^{\hat{l}}_{iu} < c^{\hat{l}}_{p(u),u}$)
and $(|V_j| + |P| \leq h)$ **then**
$P \leftarrow P \cup V_j$
$\gamma \leftarrow \gamma + c(T_j)$
end-if
$\bar{S} \leftarrow \bar{S} - \{j\}$
end-while
if $(|P| > |V_i| + 1)$ **then**
Solve to optimality the subproblem induced in G by P, and let ϕ be the value of its optimal solution.
if $(\phi < \gamma)$ **then**
Update the current solution T.
Let I and $\bar{I}$ be respectively the set of non-leaf and leaf nodes in the optimal solution of the subproblem.
$S \leftarrow (S - \bar{I}) \cup I$;
end-if
else
$S \leftarrow S - V_i$;
end-else
end-if
else
$S \leftarrow S - \{i\}$;
end-else
end-while
end-procedure

Fig. 1 Pseudo-code of the local search

to run our approach on the central node located at the edge and randomly. To run our GRASP approach on the 100 and 150 nodes instances with the central node located at the edge and randomly we shall develop extensions such as elimination tests.

The heuristic procedures were coded in C++, using compiler mingw32. CPLEX version 9.1 was used to solve all the subproblems to optimality. The 50 and 150 nodes instances were ran on a Pentium 4, 2.5 GHz, with 1 GB of RAM memory. The

Table 1 Results for the instances with 50 nodes, central node located in the center

	Best known (Raghavan 2007)			GRASP				
Instance	UB	LB	g(%)	UB	Time (s)	it	g (%)	Improved?
0	572.286	533.417	7.29	568.476	2073.375	1	6.57	yes
1	541.608	500.122	8.30	540.621	1811.016	2	8.10	yes
2	563.120	525.770	7.10	558.659	1717.453	1	6.26	yes
3	566.703	525.203	7.90	564.283	2087.125	2	7.44	yes
4	542.397	511.112	6.12	541.677	1182.516	2	5.98	yes
5	609.805	570.393	6.91	608.158	1270.328	1	6.62	yes
6	574.594	524.273	9.60	571.427	5721.984	2	8.99	yes
7	587.016	549.233	6.88	580.527	1163.516	1	5.70	yes
8	622.000	587.959	5.79	616.956	1075.734	8	4.93	yes
9	639.954	599.071	6.82	635.477	999.641	1	6.08	yes
10	560.428	528.558	6.03	557.311	1987.359	3	5.44	yes
11	597.520	563.720	6.00	592.567	1132.719	1	5.12	yes
12	633.220	593.333	6.72	630.495	1727.406	1	6.26	yes
13	545.470	501.707	8.72	541.065	1865.031	3	7.84	yes
14	570.354	533.994	6.81	565.960	2296.625	2	5.99	yes
15	561.199	531.141	5.66	560.875	1933.500	1	5.60	yes
16	581.043	541.948	7.21	577.772	1901.578	1	6.61	yes
17	576.984	539.608	6.93	570.136	1138.922	2	5.66	yes
18	612.073	566.167	8.11	605.112	1288.203	3	6.88	yes
19	609.880	572.305	6.57	608.206	1647.266	1	6.27	yes
20	564.933	515.615	9.57	561.301	2946.375	9	8.86	yes
21	595.280	555.307	7.20	593.014	2327.422	1	6.79	yes
22	570.357	533.678	6.87	565.352	2047.984	1	5.93	yes
23	575.103	527.074	9.11	573.494	2200.922	2	8.81	yes
24	629.089	584.053	7.71	623.627	1498.594	1	6.78	yes
25	559.847	539.019	3.86	554.543	2212.188	3	2.88	yes
26	575.675	532.307	8.15	570.778	899.422	4	7.23	yes
27	568.571	531.626	6.95	564.583	1076.750	1	6.20	yes
28	583.516	549.753	6.14	581.974	2180.656	1	5.86	yes
29	575.400	535.022	7.55	572.327	2128.484	1	6.97	yes
30	605.376	570.623	6.09	601.977	1419.344	1	5.49	yes
31	587.130	545.171	7.70	582.820	2425.828	12	6.91	yes
32	595.326	558.370	6.62	591.413	2155.375	1	5.92	yes
33	565.632	531.286	6.46	562.659	903.203	2	5.91	yes
34	600.291	553.032	8.55	596.988	1728.234	8	7.95	yes
35	576.653	532.293	8.33	574.187	2422.344	2	7.87	yes
36	582.251	535.656	8.70	579.590	1301.953	3	8.20	yes
37	529.314	498.100	6.27	525.880	1266.328	2	5.58	yes
38	598.707	558.825	7.14	595.329	3557.641	1	6.53	yes
39	549.914	509.313	7.97	549.045	5646.625	1	7.80	yes

Table 1 (*Continued*)

	Best known (Raghavan 2007)			GRASP				
Instance	UB	LB	g(%)	UB	Time (s)	it	g (%)	Improved?
40	559.654	521.103	7.40	554.145	1436.578	2	6.34	yes
41	582.251	544.404	6.95	578.868	2082.656	1	6.33	yes
42	588.473	541.122	8.75	581.187	1047.781	1	7.40	yes
43	585.324	547.437	6.92	581.138	1813.766	1	6.16	yes
44	585.415	558.671	4.79	584.498	1257.313	3	4.62	yes
45	585.161	543.586	7.65	581.792	1911.250	2	7.03	yes
46	581.248	546.615	6.34	575.617	1447.938	1	5.31	yes
47	626.617	574.898	9.00	622.094	3655.563	1	8.21	yes
48	586.974	542.572	8.18	579.057	1429.844	2	6.72	yes
49	559.967	532.136	5.23	559.609	1129.094	1	5.16	yes
Average	580.910	542.075	7.13	577.265	1891.729		6.45	

100 nodes instances were ran on a Athlon 64 3200+, 2 GHz, with 512 Mb of RAM memory. We ran GRASP for a number of `Max_It` equal to 15 iterations and 10 iterations for respectively the 50 nodes instances and the 100 and 150 nodes instances, since for the larger instances a larger number of subproblems are created increasing the time spent in each iteration. In the construction phase, α was chosen at random in the interval $[0.1, 0.4]$, and the parameter w, which limits the cardinality of each set in the partition, was set to 10. In the local search phase, the value of h was chosen at random from the interval $[16, 31]$.

We report in Tables 1 to 5 numerical results for each of the 250 instances considered. Tables 1, 2 and 3 correspond to the 50 nodes instances with central node located respectively in the center, at the edge and randomly. Tables 4 and 5 correspond respectively to the 100 and 150 nodes instances. In each table, the first column presents the instance identification. We then show the best known upper bound so far, the lower bound given by the linear relaxation of ESCF, and the gap with respect to the lower bound, i.e., $\frac{ub-lb}{lb}\%$. The best known upper bounds and the lower bounds were informed to us by Raghavan (2007). we report in the next columns results obtained by the proposed GRASP. We show the upper bound, the time in seconds to perform `Max_It` iterations, the iteration the best solution was found, and the gap with respect to the lower bound given by the linear relaxation of ESCF. In the last column we inform whether GRASP improved (case "yes") or not (case "no") the value of the best upper bound known so far. We display average results on the last line of each table.

Numerical results have shown the proposed hybrid strategy to be very efficient in finding good quality solutions for the MLCMST problem. GRASP using hybrid heuristic-subproblem optimization was able to improve the best known upper bounds for 247 out of 250 instances. The only instances for which the solution obtained by GRASP did not improved the best known value are the indexed by 13 and 41 with 50 nodes and central node located at the edge, and the one indexed by 34 with 100 nodes. Computational times are relatively large since the approach has to solve smaller-sized

Table 2 Results for the instances with 50 nodes, central node located at the edge

Instance	Best known (Raghavan 2007) UB	LB	g (%)	GRASP UB	Time (s)	it	g (%)	Improved?
0	1117.320	1068.715	4.55	1108.674	4939.016	1	3.74	yes
1	1152.570	1106.826	4.13	1147.926	45926.094	1	3.71	yes
2	1011.130	969.158	4.33	1007.268	17089.469	1	3.93	yes
3	1089.590	1042.383	4.53	1084.108	8939.922	3	4.00	yes
4	1124.930	1080.000	4.16	1123.234	13108.781	10	4.00	yes
5	1097.488	1052.866	4.24	1096.208	12027.750	3	4.12	yes
6	1006.158	960.042	4.80	1002.304	19471.266	1	4.40	yes
7	1046.880	994.715	5.24	1038.709	11485.844	4	4.42	yes
8	1079.360	1036.033	4.18	1077.996	15925.734	11	4.05	yes
9	1123.570	1066.439	5.36	1118.622	17105.969	1	4.89	yes
10	1122.596	1065.632	5.35	1115.708	50987.828	1	4.70	yes
11	1093.840	1042.840	4.89	1093.425	54976.344	1	4.85	yes
12	1020.390	969.542	5.24	1018.228	2256.859	2	5.02	yes
13	1158.830	1120.440	3.43	1158.830	6380.422	3	3.43	no
14	1097.010	1047.044	4.77	1093.222	8038.391	3	4.41	yes
15	1088.038	1037.810	4.84	1081.795	13872.750	1	4.24	yes
16	1048.104	1007.032	4.08	1044.068	10578.953	2	3.68	yes
17	1065.522	1014.985	4.98	1058.846	17435.656	3	4.32	yes
18	1076.144	1025.321	4.96	1073.228	9574.797	5	4.67	yes
19	1048.350	998.147	5.03	1039.356	8240.891	9	4.13	yes
20	1043.620	1008.909	3.44	1042.514	5688.469	2	3.33	yes
21	1141.432	1093.608	4.37	1138.702	7677.422	1	4.12	yes
22	1111.630	1064.362	4.44	1107.963	3579.891	14	4.10	yes
23	1024.092	976.259	4.90	1020.009	6712.328	4	4.48	yes
24	1067.950	1029.884	3.70	1064.896	20781.875	1	3.40	yes
25	1034.200	985.120	4.98	1030.405	3755.047	1	4.60	yes
26	1126.770	1082.106	4.13	1121.626	3691.234	1	3.65	yes
27	945.774	902.381	4.81	941.540	5145.250	7	4.34	yes
28	1064.890	1016.481	4.76	1059.916	13199.000	1	4.27	yes
29	1130.400	1087.191	3.97	1128.082	6261.453	12	3.76	yes
30	1047.546	1006.175	4.11	1044.712	8275.469	2	3.83	yes
31	1111.790	1071.455	3.76	1109.926	3436.719	9	3.59	yes
32	1138.346	1090.908	4.35	1135.531	34444.250	4	4.09	yes
33	1072.710	1020.903	5.07	1068.535	4951.891	15	4.67	yes
34	1030.482	983.065	4.82	1024.628	4875.438	2	4.23	yes
35	1045.550	1001.190	4.43	1039.296	6268.766	6	3.81	yes
36	1067.570	1014.803	5.20	1057.711	29760.641	5	4.23	yes
37	1006.294	959.416	4.89	1001.467	6612.141	3	4.38	yes
38	1081.320	1030.120	4.97	1080.916	4030.422	4	4.93	yes
39	1041.820	993.939	4.82	1038.405	9547.359	7	4.47	yes

Table 2 (*Continued*)

	Best known (Raghavan 2007)			GRASP				
Instance	UB	LB	g (%)	UB	Time (s)	it	g (%)	Improved?
40	1051.422	1001.262	5.01	1046.796	9388.578	10	4.55	yes
41	1051.030	1000.015	5.10	1051.428	6526.375	15	5.14	no
42	1058.582	1010.345	4.77	1052.056	6826.547	1	4.13	yes
43	1052.962	1005.704	4.70	1048.190	16615.703	3	4.22	yes
44	1018.830	974.004	4.60	1017.236	4810.297	8	4.44	yes
45	1064.190	1017.711	4.57	1062.960	6587.734	3	4.45	yes
46	1112.650	1059.193	5.05	1106.692	5963.859	10	4.48	yes
47	1119.244	1065.072	5.09	1113.767	10856.734	2	4.57	yes
48	1034.750	988.710	4.66	1028.043	9018.047	4	3.98	yes
49	1101.820	1051.821	4.75	1098.593	54296.188	9	4.45	yes
Average	1070.848	1023.538	4.60	1066.781	13141.269		4.21	

instances of an NP-Hard problem to evaluate a move. Indeed, more than 95% of the computational time effort on each instance is spent to solve exactly the subproblems. This inhibit running the method for more iterations, which could perhaps lead to further improvements in the best known upper bounds. Nevertheless, it is worth noting that, apart the instances with central node located at the edge, GRASP was able to improve the best upper bound for 187 out of 200 instances in less than one hour of CPU time. As already noted by other authors working with the CMST problem, the instances where the central node is located at the edge are in fact harder ones. This seems to also be true in the MLCMST context, as GRASP spent much more time in this type of instances because CPLEX had much more difficult in solving the subproblems to optimality.

5 Concluding remarks and extensions

The MLCMST problem possesses a structure that even simple moves defined as a reconfiguration of a subset of nodes may lead to a smaller-sized instance of the problem itself. This fact makes MLCMST a quite challenging problem for local search based algorithms. In this paper we proposed so an hybrid heuristic-subproblem optimization strategy for the MLCMST problem. Heuristics are used to build subproblems, which are in turn solved to optimality by an exact method. We embedded this scheme in a GRASP framework. The proposed GRASP has shown to be quite competitive. Considering 250 benchmark instances, it was able to improve almost all the best known upper bounds obtained by the powerful methods developed by Gamvros et al. (2006).

The main direction of further research is related to heuristic rules to speed up the search. Elimination tests and move value estimations shall be developed to reduce the number of times the local search make calls to the exact method. These extensions, if successfully managed, can enable GRASP to perform much more iterations in acceptable computational time.

Table 3 Results for the instances with 50 nodes, central node randomly located

Instance	Best known (Raghavan 2007)			GRASP				
	UB	LB	g (%)	UB	Time (s)	it	g (%)	Improved?
0	596.700	557.138	7.10	591.991	2763.495	1	6.26	yes
1	737.906	693.221	6.45	737.046	1698.13	10	6.32	yes
2	705.761	667.657	5.71	701.633	2855.06	6	5.09	yes
3	683.067	645.708	5.79	676.355	3432.59	1	4.75	yes
4	866.084	822.798	5.26	859.786	2034.48	4	4.50	yes
5	961.915	910.483	5.65	958.692	9046.31	2	5.29	yes
6	768.232	730.783	5.12	767.158	2464.86	1	4.98	yes
7	744.171	702.911	5.87	742.123	2763.63	3	5.58	yes
8	688.182	649.021	6.03	684.667	1161.94	1	5.49	yes
9	831.260	794.448	4.63	829.176	2782.61	1	4.37	yes
10	856.208	813.103	5.30	850.330	2763.50	4	4.58	yes
11	675.777	638.062	5.91	670.775	637.77	15	5.13	yes
12	718.659	672.276	6.90	715.212	4266.78	2	6.39	yes
13	795.780	752.459	5.76	790.461	1455.67	14	5.05	yes
14	791.095	745.260	6.15	786.882	3103.08	2	5.58	yes
15	874.707	828.337	5.60	872.357	1910.34	3	5.31	yes
16	704.393	663.789	6.12	702.672	969.58	2	5.86	yes
17	591.825	550.665	7.47	585.497	530.19	2	6.33	yes
18	692.904	660.101	4.97	690.706	1050.86	1	4.64	yes
19	816.991	768.450	6.32	811.374	3220.55	2	5.59	yes
20	881.706	842.081	4.71	879.342	4848.55	3	4.42	yes
21	801.536	764.734	4.81	797.794	1154.78	3	4.32	yes
22	691.108	657.499	5.11	688.662	876.70	1	4.74	yes
23	724.777	683.482	6.04	720.743	1240.91	1	5.45	yes
24	813.879	769.352	5.79	809.756	2016.31	3	5.25	yes
25	625.558	590.817	5.88	625.111	1005.27	5	5.80	yes
26	853.640	818.897	4.24	853.640	9867.516	2	4.24	yes
27	702.418	667.857	5.17	694.711	1212.938	1	4.02	yes
28	774.331	731.137	5.91	773.762	2037.203	3	5.83	yes
29	1065.210	1019.436	4.49	1062.234	6316.703	1	4.20	yes
30	717.690	679.062	5.69	712.625	2188.734	5	4.94	yes
31	736.650	689.199	6.88	729.420	1765.141	9	5.84	yes
32	977.977	933.054	4.81	973.967	4454.375	7	4.38	yes
33	976.020	929.517	5.00	970.959	2810.750	2	4.46	yes
34	796.694	754.480	5.60	792.202	2566.891	1	5.00	yes
35	552.085	522.575	5.65	551.251	977.688	2	5.49	yes
36	700.238	664.150	5.43	698.521	1407.859	1	5.18	yes
37	770.738	730.834	5.46	769.785	4535.516	4	5.33	yes
38	663.472	622.798	6.53	657.630	2455.500	1	5.59	yes
39	689.381	650.206	6.03	687.322	1205.813	5	5.71	yes

Table 3 (*Continued*)

Instance	Best known (Raghavan 2007) UB	LB	g (%)	GRASP UB	Time (s)	it	g (%)	Improved?
40	650.031	611.928	6.23	649.141	1117.766	3	6.08	yes
41	790.728	747.924	5.72	788.533	1207.609	14	5.43	yes
42	536.435	505.561	6.11	535.245	613.656	1	5.87	yes
43	720.548	687.324	4.83	719.962	1258.297	1	4.75	yes
44	627.707	590.234	6.35	621.563	605.031	1	5.31	yes
45	800.358	758.279	5.55	800.283	3250.391	5	5.54	yes
46	550.632	514.568	7.01	547.649	705.734	4	6.43	yes
47	767.506	732.944	4.72	766.440	3746.797	8	4.57	yes
48	686.307	644.523	6.48	684.083	909.844	2	6.14	yes
49	629.679	598.730	5.17	628.967	729.688	1	5.05	yes
Average	743.394	703.636	5.68	740.224	2358.306		5.22	

Table 4 Results for the instances with 100 nodes, central node located in the center

Instance	Best known (Raghavan 2007) UB	LB	g (%)	GRASP UB	Time (s)	it	g (%)	Improved?
0	1083.450	1024.191	5.79	1076.434	1801.177	7	5.10	yes
1	1117.510	1048.661	6.57	1104.727	1828.250	2	5.35	yes
2	1118.070	1051.477	6.33	1110.312	1245.278	9	5.60	yes
3	1105.440	1047.969	5.48	1096.077	1658.452	1	4.59	yes
4	1079.160	1017.385	6.07	1074.979	1995.801	10	5.66	yes
5	1117.620	1054.653	5.97	1106.464	1648.595	6	4.91	yes
6	1051.260	985.254	6.70	1043.467	1070.935	4	5.91	yes
7	1138.590	1078.650	5.56	1136.636	1594.024	9	5.38	yes
8	1112.700	1050.654	5.91	1106.025	2118.244	6	5.27	yes
9	1155.010	1077.064	7.24	1140.360	888.275	3	5.88	yes
10	1047.700	992.060	5.61	1044.391	1427.077	4	5.28	yes
11	1061.080	993.088	6.85	1048.174	3658.813	3	5.55	yes
12	1111.580	1042.317	6.65	1101.045	1479.012	3	5.63	yes
13	1084.010	1015.896	6.70	1078.415	2595.106	7	6.15	yes
14	1057.480	999.349	5.82	1053.560	1422.297	6	5.42	yes
15	1100.650	1036.539	6.19	1098.935	2158.163	6	6.02	yes
16	1092.330	1032.529	5.79	1083.318	2190.453	8	4.92	yes
17	1206.120	1134.773	6.29	1196.739	1527.039	3	5.46	yes

Table 4 (*Continued*)

Instance	Best known (Raghavan 2007)			GRASP				
	UB	LB	g (%)	UB	Time (s)	it	g (%)	Improved?
18	1111.210	1055.557	5.27	1107.420	1751.125	8	4.91	yes
19	1071.840	1010.115	6.11	1066.128	5288.727	2	5.55	yes
20	1112.400	1053.188	5.62	1103.919	2050.216	6	4.82	yes
21	1092.440	1029.058	6.16	1083.063	1186.734	10	5.25	yes
22	1036.480	974.752	6.33	1027.713	2435.408	8	5.43	yes
23	1101.340	1031.378	6.78	1092.071	1190.802	4	5.88	yes
24	1104.400	1035.416	6.66	1097.585	2037.423	3	6.00	yes
25	1095.780	1030.565	6.33	1090.636	2278.575	10	5.83	yes
26	1062.130	999.691	6.25	1059.917	2116.016	1	6.02	yes
27	1075.840	1011.056	6.41	1069.501	1594.648	3	5.78	yes
28	1077.270	1011.218	6.53	1073.943	1588.139	2	6.20	yes
29	1150.040	1076.069	6.87	1137.530	1765.426	1	5.71	yes
30	1041.360	988.261	5.37	1036.628	1693.930	8	4.89	yes
31	1072.380	1010.986	6.07	1064.187	1704.863	5	5.26	yes
32	1069.780	998.009	7.19	1060.309	1864.445	3	6.24	yes
33	1118.910	1050.361	6.53	1111.018	1654.515	3	5.77	yes
34	1072.150	1020.362	5.08	1074.551	3057.399	3	5.31	no
35	1142.252	1073.319	6.42	1128.925	1265.307	1	5.18	yes
36	1130.150	1063.698	6.25	1127.004	2542.987	4	5.95	yes
37	1126.950	1058.023	6.51	1114.458	1235.497	10	5.33	yes
38	1132.680	1071.240	5.74	1124.892	1722.432	1	5.01	yes
39	1102.900	1041.975	5.85	1097.777	2023.678	4	5.36	yes
40	1098.310	1030.333	6.60	1087.756	1189.118	7	5.57	yes
41	1081.720	1016.515	6.41	1072.136	1318.062	5	5.47	yes
42	1009.950	951.936	6.09	1008.490	1724.516	8	5.94	yes
43	1069.070	1005.234	6.35	1061.292	1722.712	5	5.58	yes
44	1105.940	1045.801	5.75	1098.635	2845.270	1	5.05	yes
45	1149.150	1073.378	7.06	1137.514	1442.134	3	5.98	yes
46	1124.670	1061.748	5.93	1122.407	2106.192	1	5.71	yes
47	1142.940	1075.421	6.28	1133.873	1241.770	7	5.44	yes
48	1075.840	1013.634	6.14	1062.411	1265.299	1	4.81	yes
49	1039.850	977.940	6.33	1034.727	1230.213	1	5.81	yes
Average	1095.056	1030.994	6.19	1087.784	1829.977		5.48	

Table 5 Results for the instances with 150 nodes, central node located in the center

	Best known (Raghavan 2007)			GRASP				
Instance	UB	LB	g (%)	UB	Time (s)	it	g (%)	Improved?
0	1580.940	1483.970	6.53	1555.086	4088.422	4	4.79	yes
1	1656.710	1569.578	5.55	1639.308	8907.453	7	4.44	yes
2	1644.040	1550.347	6.04	1624.750	7602.703	8	4.80	yes
3	1613.520	1509.925	6.86	1586.540	7786.656	6	5.07	yes
4	1659.450	1555.021	6.72	1633.248	4725.172	5	5.03	yes
5	1681.760	1583.078	6.23	1658.822	6100.688	3	4.78	yes
6	1580.190	1486.176	6.33	1560.927	5875.156	1	5.03	yes
7	1614.120	1520.713	6.14	1601.457	6128.094	6	5.31	yes
8	1616.600	1513.958	6.78	1584.621	6723.203	8	4.67	yes
9	1575.550	1477.302	6.65	1551.760	5727.875	4	5.04	yes
10	1692.590	1589.691	6.47	1670.655	7285.422	3	5.09	yes
11	1560.320	1468.010	6.29	1536.382	4121.359	7	4.66	yes
12	1630.830	1526.610	6.83	1600.248	4777.875	1	4.82	yes
13	1640.320	1548.118	5.96	1620.416	6793.125	4	4.67	yes
14	1623.600	1520.679	6.77	1596.448	6998.672	3	4.98	yes
15	1612.230	1522.013	5.93	1591.070	5340.594	8	4.54	yes
16	1575.520	1466.952	7.40	1546.311	4744.141	7	5.41	yes
17	1662.970	1566.313	6.17	1642.191	7605.047	4	4.84	yes
18	1656.130	1548.836	6.93	1627.517	6613.781	6	5.08	yes
19	1690.210	1600.140	5.63	1674.517	8642.484	8	4.65	yes
20	1540.850	1452.732	6.07	1524.373	22588.672	1	4.93	yes
21	1627.950	1542.231	5.56	1614.360	7018.938	4	4.68	yes
22	1633.970	1539.500	6.14	1615.515	4026.125	5	4.94	yes
23	1643.040	1538.718	6.78	1611.305	6657.594	2	4.72	yes
24	1664.360	1552.271	7.22	1626.808	6818.172	3	4.80	yes
25	1616.310	1512.606	6.86	1588.242	5791.656	8	5.00	yes
26	1587.260	1485.253	6.87	1564.279	11945.141	3	5.32	yes
27	1691.870	1581.592	6.97	1660.209	6437.563	6	4.97	yes
28	1599.100	1506.521	6.15	1573.735	7968.938	2	4.46	yes
29	1607.750	1502.692	6.99	1580.154	4740.234	2	5.15	yes
30	1608.570	1511.768	6.40	1585.971	7222.000	6	4.91	yes
31	1629.970	1529.406	6.58	1608.105	4494.375	1	5.15	yes
32	1640.740	1519.517	7.98	1599.565	6046.578	5	5.27	yes
33	1667.140	1572.180	6.04	1646.669	7204.375	8	4.74	yes
34	1548.240	1451.053	6.70	1523.056	6650.594	3	4.96	yes
35	1648.000	1548.540	6.42	1626.485	8050.063	7	5.03	yes
36	1551.870	1456.257	6.57	1525.583	4232.906	10	4.76	yes
37	1598.370	1509.759	5.87	1576.402	7346.703	6	4.41	yes
38	1546.380	1441.415	7.28	1519.990	12916.313	8	5.45	yes
39	1604.670	1520.043	5.57	1584.219	4259.750	3	4.22	yes

Table 5 (*Continued*)

	Best known (Raghavan 2007)			GRASP				
Instance	UB	LB	g (%)	UB	Time (s)	it	g (%)	Improved?
40	1701.160	1595.441	6.63	1673.146	11445.047	10	4.87	yes
41	1625.390	1530.982	6.17	1608.741	7715.984	9	5.08	yes
42	1654.450	1554.185	6.45	1626.899	6533.875	9	4.68	yes
43	1634.980	1539.881	6.18	1610.013	4597.219	6	4.55	yes
44	1643.670	1546.491	6.28	1613.592	5924.797	5	4.34	yes
45	1653.450	1554.087	6.39	1630.234	5386.359	9	4.90	yes
46	1669.640	1567.323	6.53	1641.708	4399.047	2	4.75	yes
47	1637.750	1534.329	6.74	1610.863	8037.734	5	4.99	yes
48	1613.150	1523.084	5.91	1597.074	6040.188	3	4.86	yes
49	1687.700	1588.236	6.26	1668.591	5103.953	10	5.06	yes
Average	1625.220	1526.607	6.44	1601.140	6827.744		4.86	

Acknowledgements We are in debt with S. Raghu Raghavan and Ioannis Gamvros for kindly making available their instances and bound values. We also wish to thank Eduardo Uchoa and Haroldo Santos for useful discussions. The second author was supported by CNPq grant 301603/2006-5, Brazil.

References

Ahuja, R.V., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms and Applications. Prentice-Hall, Englewood Cliffs (1993)

Ahuja, R.K., Orlin, J.B., Sharma, D.: Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. Math. Program. **91**, 71–97 (2001)

Amberg, A., Domschke, W., Voss, S.: Capacitated minimum spanning tree: Algorithms using intelligent search. Comb. Optim.: Theory Pract. **1**, 9–40 (1996)

Berger, D., Gendron, B., Potvin, J.Y., Raghavan, S., Soriano, P.: Tabu search for a network loading problem with multiple facilities. J. Heuristics **6**, 253–267 (2000)

Brimberg, J., Hansen, P., Lih, K.-W., Mladenović, N., Breton, M.: An oil pipeline design problem. Oper. Res. **51**, 228–239 (2003)

de Souza, M.C., Duhamel, C., Ribeiro, C.C.: A GRASP heuristic for the capacitated minimum spanning tree problem using a memory-based local search strategy. In: Resende, M.G.C., De Sousa, J.P. (eds.) Metaheuristics: Computer Decision-Making, pp. 627–657. Kluwer Academic, Dordrecht (2003)

Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. J. Heuristics **2**, 5–30 (1996)

Esau, L.R., Williams, K.C.: On teleprocessing system design. IBM Syst. J. **5**, 142–147 (1966)

Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. J. Glob. Optim. **6**, 109–133 (1995)

Festa, P., Resende, M.G.C.: GRASP: An annotated bibliography. In: Ribeiro, C.C., Hansen, P. (eds.) Essays and Surveys in Metaheuristics, pp. 325–367. Kluwer Academic, Dordrecht (2001)

Festa, P., Resende, M.G.C.: An annotated bibliography of GRASP. AT&T Labs Research Technical Report, TD-5WYSEW (2004)

Gamvros, I., Raghavan, S., Golden, B.L.: An evolutionary approach for the multi-level capacitated minimum spanning tree. Technical Research Report TR 2002-18, Institute for Systems Research, University of Maryland (2002)

Gamvros, I., Raghavan, S., Golden, B.L.: An evolutionary approach for the multi-level capacitated minimum spanning tree. In: Anandalingam, G., Raghavan, S. (eds.) Telecommunications Network Design and Management, pp. 99–124. Kluwer Academic, Dordrecht (2003)

Gamvros, I., Golden, B.L., Raghavan, S.: The multi-level capacitated minimum spanning tree. INFORMS J. Comput. **18**, 348–365 (2006)
Gavish, B.: Topological design of centralized computer networks: Formulations and algorithms. Networks **12**, 355–377 (1982)
Gavish, B.: Topological design of telecommunication networks-local access design methods. Ann. Oper. Res. **33**, 17–71 (1991)
Gouveia, L.: A $2n$ formulation for the capacitated minimal spanning tree problem. Oper. Res. **43**, 130–141 (1995)
Gouveia, L., Martins, P.: A hierarchy of hop-indexed models for the capacitated minimum spanning tree problem. Networks **35**, 1–16 (2000)
Gouveia, L., Martins, P.: The capacitated minimum spanning tree problem: revisiting hop-indexed formulations. Comput. Oper. Res. **32**, 2435–2452 (2005)
Hall, L.: Experience with a cutting plane approach for the capacitated spanning tree problem. INFORMS J. Comput. **8**, 219–234 (1996)
Martins, P.: Enhanced second order algorithm applied to the capacitated minimum spanning tree problem. Comput. Oper. Res. **34**, 2495–2519 (2007)
Martins, A.X., Souza, M.J.F., de Souza, M.C.: Modelos matemáticos para o problema da árvore geradora mínima capacitada em níveis. In: Anais do XXXVII Simpósio Brasileiro de Pesquisa Operacional, pp. 1971–1982. Gramado, Brazil (2005) (in Portuguese)
Papadimitriou, C.: The complexity of the capacitated tree problem. Networks **8**, 217–230 (1978)
Raghavan, S.: Personal communication, 2007
Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures. In: Glover, F., Kochenberger, G. (eds.) Handbooks of Metaheuristics, pp. 219–249. Kluwer Academic, Dordrecht (2003)
Rothfarb, B., Goldstein, M.: The one-terminal telpak problem. Oper. Res. **19**, 156–169 (1971)
Sharaiha, Y., Gendreau, M., Laporte, G., Osman, I.: A tabu search algorithm for the capacitated shortest spanning tree problem. Networks **29**, 161–171 (1997)
Uchoa, E., Fukasawa, R., Lysgaard, J., Pessoa, A., Poggi de Aragão, M., Andrade, D.: Robust branch-cut-and-price for the capacitated minimum spanning tree problem over a large extended formulation. Math. Program. **112**, 443–472 (2008)