Branch-and-Cut and GRASP with Hybrid Local Search for the Multi-Level Capacitated Minimum Spanning Tree Problem

Eduardo Uchoa* Túlio A.M. Toffolo⋆ Mauricio C. de Souza⋄ Alexandre X. Martins+

*Departamento de Engenharia de Produção, Universidade Federal Fluminense
Niterói, Brasil. e-mail : uchoa@producao.uff.br

⋆Departamento de Ciência da Computação, Universidade Federal de Minas Gerais
Belo Horizonte, Brasil. e-mail : tulio@toffolo.com.br

⋄Departamento de Engenharia de Produção, Universidade Federal de Minas Gerais
Belo Horizonte, Brasil. e-mail : mauricio.souza@pq.cnpq.br

+Departamento de Ciências Exatas e Aplicadas, Universidade Federal de Ouro Preto
João Monlevade, Brasil. e-mail : xmartins@decea.ufop.br

Abstract

We propose efficient algorithms to compute tight lower bounds and high quality upper bounds for the Multi-Level Capacitated Minimum Spanning Tree problem. We first develop a branch-and-cut algorithm for the problem. This algorithm is able to solve instances of medium size and to provide tight lower bounds for larger ones. We then use the branch-and-cut within GRASP to evaluate subproblems during the search. The computational experiments conducted have improved best known lower and upper bounds on benchmark instances.

Keywords: *Capacitated spanning tree, Branch-and-cut, GRASP, Subproblem optimization.*

1 Introduction

The Multi-Level Capacitated Minimum Spanning Tree (MLCMST) problem is an extension of the well-known Capacitated Minimum Spanning Tree (CMST) problem (for a comprehensive survey on the CMST probem, see Voss [6]). In the MLCMST, a feasible set of capacities is available to be installed between each pair of nodes. Thus, decisions on installing an arc to provide connection between terminal nodes and a central node can be made among different values of capacities and respective costs. 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, $l = 1, \ldots, L$, such that $0 < z^1 < z^2 < \cdots < z^L = C$, which are available to be installed on each edge $\{i, j\} \in E$ with a cost c^l_{ij}. 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 integral weight b_i is associated to each node $i \in V$. Let us designate by r the node in V which is the central node. 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 $\{p(i), i\}$.

The MLCMST problem has been treated by Gamvros et al. [1]. The authors proposed two flow-based mixed integer programming formulations and several heuristic procedures for the problem, including exponential size neighborhoods and a hybrid genetic algorithm. Recently, Martins et al. [3] proposed a GRASP using an hybrid heuristic-subproblem optimization approach for the MLCMST problem. Heuristic rules were used to define subproblems which were in turn solved exactly by employing a commercial optimization package with an integer programming model. The proposed GRASP have improved best known upper bounds for a subset of benchmark instances.

The purpose of this work is to present efficient algorithms to compute tight lower bounds and high quality upper bounds for the MLCMST problem. We propose a branch-and-cut algorithm capable to solve instances with 50 nodes, considering different locations for the central node, in a reasonable amount of time. The algorithm provides tight lower bounds for larger instances by solving relaxations on the root node. We also use the branch-and-cut within GRASP to evaluate subproblems during the construction and local search phases. This leads to a competitive algorithm to find high quality feasible solutions for the MLCMST problem.

2 Branch-and-Cut

Gouveia [2] proposed a formulation for the CMST that can be directly adapted for the MLCMST. This formulation works over a directed graph $G_D = (V, A)$, where A has a pair of opposite arcs (i, j) and (j, i) for each edge $e = \{i, j\} \in E$, excepting edges $\{r, i\}$, which are transformed into a single arc (r, i). The solution must be an arborescence having directed paths from node r to all the remaining nodes. The formulation requires the following assumptions: (i) $z^1 = 1$ and (ii) capacities increase from 1 to z^L by unitary increments. The cases in which conditions (i) and (ii) do not hold can be handled by introducing artificial capacities. The cost associated to an artificial capacity is the same of the first available capacity greater than the artificial one. Let binary variables x_a^d indicate that arc $a = (i, j)$ belongs to the optimal arborescence and that the total weight of the nodes in the sub-arborescence rooted in j is exactly d. Let $l(d)$ denote the smaller l such that $z^l \geq d$, and let $\delta^-(i) \subseteq A$ (resp. $\delta^+(i) \subseteq A$) the set of incoming (resp. outgoing) arcs of node $i \in V$.

$$\text{Minimize} \quad \sum_{a \in A} \sum_{d=1}^{C} c_a^{l(d)} x_a^d \tag{1a}$$

S.t.

$$\sum_{a \in \delta^-(i)} \sum_{d=1}^{C} x_a^d = 1 \qquad (\forall\, i \in V \setminus \{r\}), \tag{1b}$$

$$\sum_{a \in \delta^-(i)} \sum_{d=1}^{C} d x_a^d - \sum_{a \in \delta^+(i)} \sum_{d=1}^{C} d x_a^d = b_i \qquad (\forall\, i \in V \setminus \{r\}), \tag{1c}$$

$$x_a^d \in \{0, 1\} \qquad (\forall\, a \in A;\ d = 1, \ldots, C). \tag{1d}$$

This formulation was already used in [3] in order to solve (using only standard CPLEX routines) small-sized MLCMST that were generated as subproblems in the GRASP heuristic. This paper proposes enhancing this formulation with powerful cuts, so the resulting branch-and-cut algorithm can solve larger instances or at least provide significantly better lower bounds in its root node.

2.1 Extended Capacity Cuts

For any set $S \subseteq V \setminus \{r\}$, define $b(S) = \sum_{i \in S} b(i)$. Summing equations (1c) over S, one gets:

$$\sum_{a^d \in \delta^-(S)} d x_a^d - \sum_{a^d \in \delta^+(S)} d x_a^d = b(S). \tag{2}$$

An *Extended Capacity Cut* over S is any inequality valid for the polyhedron given by the convex hull of the 0-1 solutions of (2) (i.e., the solutions of a quite particular equality-constrained knapsack problem).

In practice, such inequalities can be separated in a fast way by multiplying equations (2) by suitable multipliers and applying integer rounding. This separation procedure is described in Uchoa et al. [5], where the same inequalities were used in a branch-cut-and-price algorithm.

2.2 Fenchel Cuts over Sets of Size 2

Let $S = \{u, v\} \subset V \setminus \{r\}$ be a set of size 2. Define the arc-set $A(S)$ as $\delta^-(S) \cup \delta^+(S) \cup \{(u,v),(v,u)\}$. Let $x(S)$ the subset of the variables x_a^d where $a \in A_S$. Let $P(S)$ be the set composed by the 0-1 incidence vectors that correspond to the possible integral values for the variables in $x(S)$. Let $\bar{x}$ be the current fractional solution in the branch-and-cut and $\bar{x}(S)$ its restriction to $A(S)$. In a similar way, let α be a vector of coefficients associated to the x variables and $\alpha(S)$ its restriction to $A(S)$. If the solution of the following linear program over variables $\alpha(S)$ yields $z^* > 1$, then $\alpha.x \leq 1$ (the positions of α not in $\alpha(S)$ are completed with zero) is a valid cut.

$$\text{Maximize } z = \quad \bar{x}(S).\alpha \tag{3a}$$

$$\text{S.t.}$$

$$p.\alpha \leq 1 \quad (\forall\, p \in P(S)), \tag{3b}$$

$$\alpha \geq 0. \tag{3c}$$

The separation of such kind of Fenchel cuts is quite practical because one can further restrict $A(S)$ to the arcs with positive value in the current fractional solution. In this way, the size of the sets $P(S)$ is not too large. Moreover, only sets S where $\sum_{d=1}^{C}(\bar{x}_{uv}^d + \bar{x}_{vu}^d) > 0$ need to be considered. As far as we know, those cuts were never used before for capacitated spanning tree problems.

3 GRASP

Our GRASP employs the heuristic rules proposed by Martins et al. [3] to generate smaller-sized MLCMST subproblems. These subproblems are independently solved during the search by the proposed branch-and-cut algorithm, c.f. Section 2.

3.1 Construction Phase

In the construction phase, we first use a greedy randomized heuristic to do a partition of $V - \{r\}$ in R_k, $k = 1, \ldots, K$, subsets. The cardinality of each subset R_k is limited by a parameter $w \geq z^L$. Initially, $R_k = \emptyset$ for $k = 1, \ldots, K$, and k is set to 1. A Restricted Candidate List (RCL) comprises nodes whose incorporation to R_k results in the smallest incremental cost according to Prim's algorithm to compute a minimum spanning tree. The node to be inserted in R_k is then randomly selected from RCL. When w nodes are inserted in R_k, we increment k and proceed until a partition of $V - \{r\}$ is done. Subproblems consist of K independent MLCMST instances defined each on a subgraph induced in G by $R_k \cup \{r\}$, $k = 1, \ldots, K$. Then, we apply the proposed branch-and-cut to solve each of the K subproblems to optimality.

3.2 Local Search Phase

In the local search phase, we try to re-arrange nodes of different components connected to r. Given a feasible tree T, a neighbor is obtained by (i) defining a subgraph $\bar{G}$ of G, and (ii) solving a MLCMST subproblem on $\bar{G}$. Considering the forest composed of Q connected components when removing node r and its adjacent edges from T, the subgraph $\bar{G}$ is induced in G by $\bar{V} = \{r\} \cup_{q \in \Phi} V_q$ where $\Phi \subset \{1, \ldots, Q\}$ and V_q is the set of nodes of component q. A neighbor solution is obtained by re-arranging the components whose indexes belong to Φ, leaving the other components unchanged. This kind of move leads to the need of solving a smaller-sized MLCMST instance in subgraph $\bar{G}$ in the worst-case. We use heuristic rules in selecting components to form $\bar{G}$, avoiding the need of evaluating all possible moves. To evaluate a considered move, we apply the proposed branch-and-cut to solve to optimality the subproblem on $\bar{G}$.

4 Computational Results

We report numerical results on a subset of benchmark instances introduced in the literature by Gamvros et al. [1]. These are graphs with 50, 100, and 150 nodes plus the central node, divided into three groups: central node located in the center, at the edge, or randomly. Nodes with unitary demand are randomly distributed in a 40×40 square grid. Capacity values are $z^1 = 1$, $z^2 = 3$ and $z^3 = 10$; the cost c^1_{ij}, $\{i, j\} \in E$ is equal to Euclidean distance (not rounded), and then $c^2_{ij} = 2c^1_{ij}$ and $c^3_{ij} = 3c^1_{ij}$. We consider the first 5 instances out of 50 generated for each group resulting in a total of 45 instances.

The procedures were coded in `C++`, compiler `gcc` 4.2.3, and CPLEX 10.2 was used. Experiments were perfomed on a machine Intel Core 2 Duo 2.5Ghz with 4GB de RAM running LINUX. Table 1 presents optimal values obtained with the branch-and-cut for instances with 50 nodes. Instances with central node in the center are identified by "c", at the edge by "e", and randomly by "r". Computational times are in seconds. Tables 2 and 3 present upper (UB) and lower (LB) bounds for instances with 100 and 150 nodes respectively. First, the best known upper and lower bounds, with respective percentage gaps, are reported. Then, upper bounds obtained by running 10 iterations of GRASP with subproblems up to 50 nodes, and lower bounds obtained at the root node of the branch-and-cut are reported. The best known upper bounds for instances with central node in the center were obtained by Martins et al. [3]. For instances with central node at the edge or randomly located, the best known upper bounds were obtained by Gamvros et al. [1]. The best known lower bounds for all instances with 100 and 150 nodes were also obtained by Gamvros et al. [1]. The upper and lower bounds obtained by Gamvros et al. [1] were informed to us by Raghavan [4]. Computational results obtained with the approaches proposed in this work improved the best values for both upper and lower bounds, and percentage gaps are now significantly reduced.

inst.	opt*	time
50-0c	568.48	52.01
50-1c	540.62	65.08
50-2c	558.66	29.44
50-3c	564.28	52.12
50-4c	541.68	62.37

inst.	opt*	time
50-0e	1108.67	171.09
50-1e	1147.73	128.27
50-2e	1007.27	125.80
50-3e	1084.11	222.10
50-4e	1123.23	557.88

inst.	opt*	time
50-0r	591.99	74.07
50-1r	737.05	109.47
50-2r	701.63	67.94
50-3r	676.35	32.95
50-4r	859.79	67.03

Table 1: Optimal values for instances with 50 nodes.

References

[1] I. Gamvros, B.L. Golden, and S. Raghavan, *The multi-level capacitated minimum spanning tree*, INFORMS Journal on Computing **18** (2006) pp. 348–365.

[2] L. Gouveia, *A 2n formulation for the capacitated minimal spanning tree problem*, Operations Research **4** (1995) pp. 130–141.

[3] A.X. Martins, M.C. de Souza, M.J.F. Souza, T.A.M. Toffolo, *GRASP with Hybrid Heuristic-Subproblem Optimization for the Multi-Level Capacitated Minimum Spanning Tree Problem*, Journal of Heuristics, to appear, DOI 10.1007/s10732-008-9079-x.

[4] S. Raghavan, personal communication, 2007.

[5] E. Uchoa, R. Fukasawa, J. Lysgaard, A. Pessoa, M. Poggi de Aragão, and D. Andrade, *Robust branch-cut-and-price for the capacitated minimum spanning tree problem over a large extended formulation*, Mathematical Programming **112** (2008) pp. 443–472.

	Best Known			GRASP + B&C				
inst	UB	LB	g(%)	UB	time	LB	time	g(%)
100-0c	1076.434	1024.191	5.10	1075.429	3527	1073.745	266	0.16
100-1c	1104.727	1048.661	5.35	1102.352	3760	1099.220	199	0.28
100-2c	1110.312	1051.477	5.60	1108.356	3925	1103.450	1152	0.44
100-3c	1096.077	1047.969	4.59	1096.077	3138	1093.370	201	0.25
100-4c	1074.979	1017.385	5.66	1073.830	3754	1069.510	208	0.40
100-0e	2185.180	2102.977	3.91	2168.097	6709	2158.867	533	0.43
100-1e	2015.960	1943.439	3.73	2007.625	6143	2000.413	1392	0.36
100-2e	2120.680	2054.838	3.20	2111.526	5481	2103.561	326	0.38
100-3e	1993.730	1928.662	3.37	1989.065	5963	1978.623	601	0.53
100-4e	2037.460	1966.557	3.61	2032.877	6301	2020.175	1136	0.63
100-0r	1606.570	1533.735	4.75	1594.350	6271	1588.088	438	0.39
100-1r	1893.580	1816.662	4.23	1885.063	6914	1875.209	490	0.53
100-2r	1488.340	1418.726	4.91	1476.980	5555	1473.078	277	0.26
100-3r	1175.050	1108.194	6.03	1166.972	4039	1159.709	418	0.63
100-4r	1166.970	1101.724	5.92	1155.714	3474	1149.273	188	0.56

Table 2: Upper and lower bounds for instances with 100 nodes

	Best Known			GRASP + B&C				
inst	UB	LB	g(%)	UB	time	LB	time	g(%)
150-0c	1555.086	1483.970	4.79	1550.382	7894	1541.393	728	0.58
150-1c	1639.308	1569.578	4.44	1634.420	5855	1627.047	1004	0.45
150-2c	1624.750	1550.347	4.80	1620.860	5924	1611.212	1086	0.60
150-3c	1586.540	1509.925	5.07	1578.877	6347	1569.488	1130	0.60
150-4c	1633.248	1555.021	5.03	1626.312	5826	1617.757	977	0.53
150-0e	3043.190	2932.044	3.79	3003.718	9120	2992.415	692	0.38
150-1e	3130.600	3017.356	3.75	3099.175	10151	3082.438	762	0.54
150-2e	3074.820	2969.406	3.55	3043.186	9807	3028.971	501	0.47
150-3e	3049.500	2944.365	3.57	3025.921	9360	3008.361	743	0.58
150-4e	3057.150	2930.994	4.30	3012.149	9329	2993.525	830	0.62
150-0r	2418.800	2303.125	5.02	2374.978	10067	2363.651	353	0.48
150-1r	2122.230	2020.047	5.06	2087.433	8119	2076.390	299	0.53
150-2r	2243.700	2135.843	5.05	2216.420	8431	2201.894	323	0.66
150-3r	2180.740	2076.635	5.01	2147.615	8081	2135.203	320	0.58
150-4r	2236.040	2128.817	5.04	2206.680	8877	2195.203	367	0.52

Table 3: Upper and lower bounds for instances with 150 nodes

[6] S. Voss, *Capacitated minimum spanning trees*, **In:** C.A. Floudas and P.M. Pardalos (eds.) Encyclopedia of Optimization, Vol. 1, 2nd Edition, Springer, Berlin, 2008 pp. 225–235.