Branch-and-Cut and 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§ Ricardo Fukasawa ¶

May 29, 2020

Abstract

We propose 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, introducing some new features: (i) the exact separation of cuts corresponding to some master equality polyhedra found in the formulation; (ii) the separation of Fenchel cuts, solving LPs considering all the possible solutions restricted to small portions of the graph. We then use that branch-and-cut within a GRASP that performs moves by solving to optimality subproblems corresponding to partial solutions. The computational experiments were conducted on 450 benchmark instances from the literature. Numerical results show improved best known upper bounds for almost all instances that could not be solved to optimality.
Keywords : Network Design, Branch-and-Cut, Fenchel Cuts, MIP based local search.

1 Introduction

The Multi-Level Capacitated Minimum Spanning Tree (MLCMST) problem is an extension of the well-known Capacitated Minimum Spanning Tree (CMST) problem. The CMST problem consists 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

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

†Departamento de Computação, Universidade Federal de Ouro Preto, Ouro Preto, MG, Brazil. e-mail : tulio@toffolo.com.br

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

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

¶Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada. e-mail : rfukasaw@math.uwaterloo.ca

other nodes, called terminals, is bounded by a capacity C on each edge. Esau and Williams [9] introduced the CMST, and since then a lot of research on both exact and heuristic methods to tackle the problem has been conducted, see for instance Ahuja et al. [1, 2], Amberg et al. [3], Hall [17], Gouveia [14], Gouveia and Martins [15, 16], Martins [18], Sharaiha et al. [23], de Souza et al. [24], and Uchoa et al. [25]. The reader can also address the survey by Voss [26].

In the MLCMST case, a feasible set of capacities is available to be installed between each pair of nodes. This means that instead of installing a capacity C on every edge of the tree, as happens in the CMST, a choice 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}. 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 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 recently treated in the literature by Gamvros et al. [12, 13], and by Martins et al. [19]. Gamvros et al. [13] proposed two flow-based mixed integer programming formulations and several heuristic procedures for the problem, including exponential size neighborhoods and a hybrid genetic algorithm. Martins et al. [19] proposed a GRASP hybridized with an integer programming model. The heuristic defines subproblems that are solved by a commercial optimization package over that IP. That hybridized GRASP improved the best known upper bounds for several 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 of solving instances with 50 nodes (and even some instances with 100 nodes) in a reasonable amount of time. Moreover, the relaxations solved at the root node of the algorithm provide very tight lower bounds (always less than 1%) for larger instances with up to 150 nodes. We then use the branch-and-cut within a GRASP similar to that in [19], optimizing subproblems during the construction and local search phases. Since the new branch-and-cut is substantially more effective than the pure IP model, the resulting algorithm is able to find solutions better than those found by the former algorithm in a consistent way.

The paper is structured as follows. In the next section we describe the branch-and-cut algorithm, that differs from that in Uchoa et al. [25] by having new strong separation procedures. First, the so-called Extended Capacity Cuts that were only separated heuristically in [25] are now also separated exactly, using a characterization of the master equality polyhedron found recently by Dash, Fukasawa and Günlük [8].

Second, it uses a newly defined family of Fenchel cuts separated by considering the possible partial solutions over small parts of the graph. Then, in Section 3, we describe how the GRASP employs the branch-and-cut to solve subproblems generated in the construction and local search phases. Section 4 is devoted to report computational experiments conducted on all the 450 larger benchmark instances introduced in the literature by Gamvros et al. [13]. We present optimal solution values of all the 150 instances with 50 nodes and all the 50 instances with 100 nodes and the root in a central position. We then show significantly reduced gaps for all the remaining 250 instances, with 100 and 150 nodes.

2 Branch-and-cut

Gouveia [14] 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, in other words, $z^i = i$ for $i = 1, \ldots, L$. 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 as 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. For any $S \subset V$, let $\delta^-(S) := \{(i, j) \in A : i \notin S, j \in S\}$ and $\delta^+(S) := \{(i, j) \in A : i \in S, j \notin S\}$ denote the set of incoming (resp. outgoing) arcs of set S. For simplicity, let $\delta^+(i) := \delta^+(\{i\})$ and $\delta^-(i) := \delta^-(\{i\})$. Then a formulation for the MLCMST is

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

S.t.

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

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

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

This formulation was already used in [19] in order to solve (using CPLEX's built in branch-and-bound solver) small-sized MLCMST problems 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 at the 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:

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

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). The *Homogeneous Extended Capacity Cuts* (HECCs) are a subset of the ECCs where all entering variables with the same capacity have the same coefficients, the same happening with the leaving variables. For a given set S, define aggregated variables y^d and z^d as follows:

$$y^d = \sum_{a^d \in \delta_d^-(S)} x_a^d \qquad (d = 1, \ldots, C), \tag{3}$$

$$z^d = \sum_{a^d \in \delta_d^+(S)} x_a^d \qquad (d = 1, \ldots, C). \tag{4}$$

Equation (2) implies that:

$$\sum_{d=1}^{C} dy^d - \sum_{d=1}^{C} dz^d = b(S). \tag{5}$$

For each possible pair of values of C and $D = b(S)$, we may define the polyhedron $P(C, D)$ induced by the non-negative integral solutions of (5), a very particular kind of equality constraint where all integral coefficients in the range $[-C, \ldots, C]$ are present. The inequalities that are valid for those polyhedra are HECCs. In practice, given a set S, a good heuristic to separate such inequalities is by multiplying equation (5) by suitable multipliers and applying integer rounding. The separation procedure used to select candidate sets S and multipliers is fully described in Uchoa et al. [25], where the same inequalities were used in a branch-cut-and-price algorithm.

Recently, Dash, Fukasawa and Günlük [8] performed a deep study of polyhedra $P(C, D)$, which they called the *Master Equality Polyhedra*. In particular, they give a pseudo-polynomial characterization of the polar of such polyhedra. This means that one can separate a point from $P(C, D)$ by solving a linear program of pseudo-polynomial size ($O(C)$ variables and $O(C^3)$ constraints).

In this work, we use that result to significantly improve the separation of HECCs. We still use the same procedure of [25] to select candidate sets S. But now, when the heuristic separation over $P(C, D)$ by rounding fails, we call an exact separation procedure by LP solving.

2.2 Fenchel cuts over the neighborhood of two nodes

The so-called *Fenchel cuts* for integer programming were introduced by Boyd in the early nineties [7]. These cuts are characterized for being separated by solving a linear program where the variables correspond to the coefficients of the desired cut, maximizing the violation with respect to the current fractional solution subject to not cutting any integer feasible solution. Of course, Fenchel cut separation can not be

applied to a whole IP, separating a single cut would be more expensive than solving the original problem. In practice, those cuts are separated with respect to substructures present in the IP, typically knapsack-like constraints [6, 11, 21, 5, 22]. Applegate et al. [4] separated Fenchel cuts for the TSP (called *local cuts*) by performing node contractions that shrink the original graph into a much smaller graph and by considering the solutions of the graphical TSP (a relaxation of the TSP that allows multiple visits to a node) on the shrunk graph. In this paper, we separate Fenchel cuts for the MLCMST by considering small parts of a fractional solution.

Let $S = \{u, v\} \subset V \setminus \{r\}$ be a set containing two non-root nodes. Define the neighborhood of S as the arc-set $A(S) = \delta^-(S) \cup \delta^+(S) \cup \{(u, v), (v, u)\}$ and $x(S)$ as 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 possible integral values for the variables in $x(S)$ and are maximal with respect to the number of values 1. If $\bar{x}$ is the current fractional solution in the branch-and-cut, we denote by $\bar{x}(S)$ 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.

$$\text{Maximize } z = \quad \bar{x}(S).\alpha(S) \tag{6a}$$

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

$$p.\alpha(S) \leq 1 \quad (\forall\, p \in P(S)), \tag{6b}$$

$$0 \leq \alpha \tag{6c}$$

The separation of such kinds 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 number of solutions in $P(S)$ to be considered is not too large. Moreover, only sets $S = \{u, v\}$ where $\sum_{d=1}^{C}(\bar{x}_{uv}^d + \bar{x}_{vu}^d) > 0$ need to be considered.

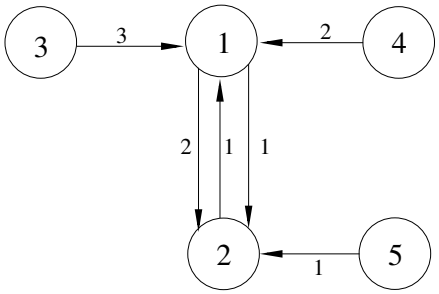

Figure 1: Partial fractional solution over set $A(\{1, 2\})$, all arcs have value 1/3.

For example, Figure 1 depicts the arcs with positive value in a fractional solution restricted to $A(S)$, where $S = \{1, 2\}$. All those arcs have a value of 1/3, the numbers

next to the arrows are the d indices. We solve the following LP:

$$\begin{aligned}
\text{Maximize } z = \quad & 1/3(\alpha^1_{12} + \alpha^2_{12} + \alpha^1_{21} + \alpha^3_{31} + \alpha^2_{41} + \alpha^1_{52}) && \text{(7a)}\\
\text{S.t.} \quad & \\
& \alpha^1_{12} + \alpha^3_{31} \leq 1 && \text{(7b)}\\
& \alpha^1_{12} + \alpha^2_{41} \leq 1 && \text{(7c)}\\
& \alpha^2_{12} + \alpha^3_{31} \leq 1 && \text{(7d)}\\
& \alpha^3_{31} + \alpha^1_{52} \leq 1 && \text{(7e)}\\
& \alpha^2_{41} + \alpha^1_{52} \leq 1 && \text{(7f)}\\
& \alpha^1_{21} \leq 1 && \text{(7g)}\\
& 0 \leq \alpha. && \text{(7h)}
\end{aligned}$$

We remark that the partial solutions corresponding to inequalities (7b-7f) are maximal, it is not possible to have solutions where more than 2 of the 6 considered variables have value 1. The partial solution with only x^1_{21} having value 1 is also maximal, this solution corresponds to the inequality (7g). Solving that LP, one gets $z^* = 4/3$ and the following violated inequality:

$$x^1_{12} + x^2_{12} + x^1_{21} + x^1_{52} \leq 1$$

In this particular case, the derived Fenchel cut has a clear structure: a clique cut. In fact, it could be lifted to a stronger cut by including positive coefficients for some variables in $A(S)$ with $\bar{x}^d_a = 0$, that were not considered in the separation. For example, x^2_{21} (and several other variables) could have their coefficients increased to 1 by noticing that this variable is incompatible with all the variables already with coefficient 1. We decided not to perform any lifting. The difficulty of lifting lies in the fact that general Fenchel cuts do not have a clear structure to help on the lifting. For example, Figure 2 depicts the support graph of a cut found over a certain fractional solution of a benchmark instance. Again, the numbers next to the arrows represent the d indices. The cut is (we multiplied the actual $\alpha(S)$ in the LP solution by 4 in order to have integral coefficients):

$$3x^1_{12} + 3x^2_{12} + x^1_{13} + x^3_{13} + x^1_{21} + x^1_{23} + x^3_{23} + x^3_{25} + x^1_{41} + 2x^3_{62} \leq 4.$$

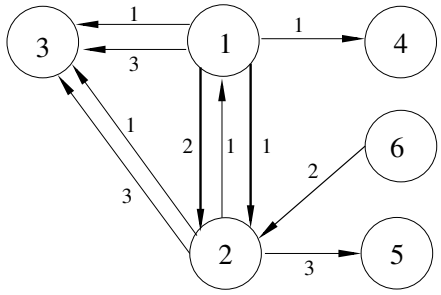

Figure 2: Support graph of a Fenchel cut.

In cases like that (there are cuts that are much more complicated), we do not know how to find valid lifting coefficients for certain variables, except by enumerating all partial solutions that include that variable. This would be expensive, the number of partial solutions to be considered increases exponentially with the number of lifted variables.

It is interesting that we did not experience any numerical difficulty with our Fenchel cuts on the MLCMST problem (for example, those kind of problems were reported in the Generalized Assignment Problem [5]). In our case the cuts may be complex, but always have small integral coefficients (after multiplying by a suitable factor), usually up to 5 and rarely more than 8. As will be shown in the experimental results, the Fenchel cuts were able to close a significant part of the remaining integrality gap (after the separation of ECCs) in reasonable times.

As far as we know, Fenchel cuts were never used before in network design problems.

3 GRASP

Our GRASP employs the heuristic rules proposed by Martins et al. [19] 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. We also use a verification routine to check if a new subproblem generated has been already investigated. The purpose of such a routine is to avoid recomputations.

GRASP is a multi-start metaheuristic that has been widely used to obtain good quality solutions for many combinatorial problems (see [10]). 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 minimum is reached. The procedure returns the best solution found after `Max_It` iterations.

3.1 Construction phase

The construction phase proposed by Martins et al. [19] uses a greedy randomized heuristic to partition $V - \{r\}$ into subsets $R_1, \ldots, R_K$. 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. While $|R_k| < w$, the procedure inserts a node not been allocated to any set in the partition under construction to R_k. A Restricted Candidate List (RCL) comprises nodes whose insertion in R_k results in the smallest incremental cost according to Prim's algorithm to compute a minimum spanning tree. Let d_j, for a node j not been allocated to any set in the partition, be a label defined as $d_j = \min\{c^1_{ij} : i \in R_k\}$. Then, d_{min} and d_{max} denote respectively the minimum and the maximum values of d_j. Given a parameter $\alpha \in [0, 1]$, a node j not been allocated belongs to RCL if $d_j \leq d_{min} + \alpha(d_{max} - d_{min})$. The node to be inserted in R_k is randomly selected from those in the RCL. When w nodes are inserted in R_k, the heuristic increments k and proceeds until a partition of $V - \{r\}$ is completed. 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

Martins et al. [19] developed a local search that tries 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 $\bar{G}$. Let us assume that the forest obtained by removing node r and its adjacent edges from T is composed of B connected components. A subproblem is formed with a subgraph $\bar{G}$ induced in G by $\bar{V} = \{r\} \cup_{b \in \Phi} V_b$ where $\Phi \subset \{1, \ldots, B\}$ and V_b is the set of nodes of component b. A neighbor solution is obtained by re-arranging in a optimal manner the components whose indices belong to Φ, leaving the other components unchanged. The cardinality of $\bar{V}$ is limited by a parameter h. Every time a new subproblem is generated the parameter h is chosen at random from the interval $[\underline{h}, \bar{h}]$, where $\underline{h}$ and $\bar{h}$ are positive integers.

A non-leaf node i in T is chosen to be the "reference" to form a subproblem. Let V_i be the set of nodes in the component containing node i. Initially, $\bar{V}$ is set to $V_i \cup \{r\}$. The procedure looks for a candidate node u belonging to a component different to V_i in T that could be connected to i at a smaller cost. That is, a node u such that $c^{\hat{l}}_{iu} < c^{\hat{l}}_{p(u)u}$ where $\hat{l}$ is the capacity on edge $\{p(u), u\}$ in T. Then, $\bar{V}$ is enlarged to $\bar{V} \cup V_u$ if the cardinality of the resulting set does not exceed h. The procedure continues trying to enlarge the subproblem until either the cardinality of $\bar{V}$ is h or all candidate nodes have been tested.

This kind of move leads to the need of solving a smaller-sized MLCMST instance in subgraph $\bar{G}$ in the worst-case. To evaluate a considered move, we apply the proposed branch-and-cut to solve to optimality the subproblem. We propose a routine to check if a move under consideration leads to a subproblem already investigated during the search, since evaluating a move is the most time consuming part of the heuristic. The verification routine works as follows. Let us assume P subproblems been solved during the search, each one of them labeled from 1 to P. For each node i we store the labels of the subproblems in which node i appears. Every time a subset $\bar{V} \subseteq V$ is selected to form a new subproblem, we check for the subproblem labels in common to all nodes in $\bar{V}$. If there is not a label in common, the subproblem is a new one, and it is solved to optimality. If there is a label p in common to all nodes in $\bar{V}$, we need to check if the set of nodes forming subproblem p is a subset of $\bar{V}$. To do this, for each subproblem being solved we store its number of nodes. If the number of nodes of subproblem p is less than the cardinality of $\bar{V}$, the subproblem is a new one, and it is solved to optimality. In such a case the procedure continues from the optimal tree of subproblem p avoiding recomputation, since we store for each subproblem its optimal solution. Otherwise subproblem p had $\bar{V}$ as the set of nodes, and hence we can simply use the stored optimal solution with minimal computational cost.

4 Computational results

Gamvros et al. [13] introduced in the literature a set of benchmark instances for the MLCMST problem. The larger instances have 50, 100, and 150 terminal nodes - besides the central node - randomly distributed in a 40×40 square grid. All demands are unitary. 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 capacity value is equal to the Euclidean distance between nodes i and j (not rounded), and the cost of the second and third capacities are respectively $c^2_{ij} = 2c^1_{ij}$ and $c^3_{ij} = 3c^1_{ij}$. There are three instance classes for each size, according to the location of the central node: in the center, at the edge, and randomly located. Gamvros et al. [13] generated a total of 450 larger instances, divided into the following 9 series of 50 instances with the same size and class: 50c, 50e, 50r, 100c, 100e, 100r, 150c, 150e, and 150r. Martins et al. [19] used the 250 instances from series 50c, 50e, 50r, 100c, and 150c on their experiments. In this work, all the 450 larger instances were used in the experiments. All algorithms were coded in `C++`, and compiled on `gcc/g++` version 2.4. The branch-and-cut algorithm was actually implemented using the *cut callback* feature present in CPLEX version 10.2.0 callable library, separating the cuts described in Section 2. The branch-and-cut uses default CPLEX parameters. We ran all the experiments on a Core 2 Quad 2.5 GHz with 4Gb of RAM memory running Linux Ubuntu 8.04.

Table 1 is aimed at showing the impact of each family of cuts on improving the lower bounds at the root node of the branch-and-cut. For series 50c, 50e, 50r, and 100c (where proven optimal solutions could be found for all instances), we report average integrality gaps: for the linear relaxation of formulation (1) (LP); for also adding HECCs separated by the same rounding heuristic used in [25] (+HECC Heu.); for also adding HECCs found by the exact separation provided by the results in [8] (+HECC Ex.); for also adding the Fenchel cuts proposed in this paper (+Fenchel). For the last case, that includes all cuts in this paper, the maximum gap is also provided. It can be seen that the last two new families of cuts are quite effective, typically dividing the remaining integrality gap by a factor of 3.

Tables 2 and 3 report the optimal solution values (within a precision of 3 decimal places) and the time taken by the full branch-and-cut algorithm (without using any external upper bound) to solve all instances in series 50c, 50e, 50r, and 100c. Each table is arranged into two blocks of columns, corresponding to a series. The first column of a block identifies each particular instance in the series, numbered from 0 to 49. Then, the optimal values and computational times in seconds are shown in the second and third columns respectively. Some remarks about those results:

- The branch-and-cut was able to quickly solve all instances from 50c series. As already observed in the CMST case (see for instance [15]) instances where the central node is located at the edge are harder. Those where the central node is randomly positioned have an intermediate difficulty. Nevertheless, the maximum time to solve any instance with 50 nodes is only 461.3 seconds. The standard CPLEX MIP solver is much slower on solving those instances using only formulation (1), the average times are 145.9 (50c), 1305.6 (50e), and 210.9 (50r). The maximum time is 5601.1 seconds.

- The branch-and-cut could also solve all instances from the 100c series, but this takes significantly more time, 6780 seconds in average. The standard CPLEX MIP could not solve any instance from the 100c series, it gets out of memory after several hours of processing time.
- The branch-and-cut algorithm can not solve most of the instances from the remaining series in reasonable computing times.

In this context, the GRASP heuristic using the branch-and-cut to solve subproblems having up to 51 nodes becomes an interesting alternative to find high quality solutions for those larger/harder instances. The original GRASP in [19] only solved subproblems with up to 31 nodes, since their general MIP solver would take too much time to handle larger subproblems. We remark that the GRASP usually generates subproblems where the central node is located at the edge, even when this central node has a central position with respect to the complete instance. As mentioned before, these types of problems are typically harder to solve, so having a more efficient code for solving the subproblems is a very important tool.

Tables 4 to 8 report upper and lower bounds for all instances in series 100e, 100r, 150c, 150e, and 150r. The new upper bounds were obtained by the GRASP employing the branch-and-cut in the construction and local search phases. We ran GRASP for a number of `Max_It` equal to 10 iterations, but also report results for a single iteration. In the construction phase, α was chosen randomly 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 $[16, 51]$. The new lower bounds are given by running only the root node of the branch-and-cut. Columns in those tables have the following meaning. We first report the best upper bounds (UB) and lower bounds (LB) known before this paper, and the respective gaps, i.e., $\frac{UB-LB}{LB}\%$. The previously best known upper bounds for the 150c series are taken from Martins et al. [19]. The other previous best known upper bounds and all previous best lower bounds were obtained by Gamvros et al. [13, 20]. The next four columns are the results obtained by the GRASP: the upper bounds after 1 and 10 iterations, then the respective running times in seconds. The following columns are the results for solving the root node of the branch-and-cut over the complete instance – the lower bound and the running time in seconds. In the last column, we present the gap between the lower bound and the best upper bound found. Some comments about those results:

- The upper bounds provided by the full GRASP with the branch-and-cut are certainly very good, the maximum gap over all those large/harder 250 instances was less than 0.88%. However, it is also clear that this heuristic is quite expensive. Running times of the version with 10 iterations range between 1 to 6 hours. In cases where those times are unacceptable, a possible strategy is parallelizing the algorithm using more than one processor (or core), noting that GRASP iterations are independent.
- Even if one cannot parallelize the algorithm or wait a few hours for its results, the experiments have shown that a single iteration of the GRASP (when the algorithm basically is reduced to an hybrid local search) already gives fairly good solutions. The largest gap observed using that upper bound was 1.25%.

The average time of a single GRASP iteration over the instances with 150 nodes is 986 seconds. For those instances, Gamvros et al. actually chose not to use their best heuristic since it became too time consuming. Their running times for this faster, but potentially worse heuristic is around 1000 seconds, and therefore roughly comparable to the single GRASP iteration (albeit in a slower machine, a dual-processor Pentium II 1GHz with 512MB RAM). Nonetheless, the upper bounds obtained after a single GRASP iteration are better than the previous best known ones in 64% of the 150c instances and in 100% of the 150e and 150r instances. Hence, taking roughly the same amount of time, we were able to improve significantly on their results.

- Due to the running times, the lower bounds from the Gamvros et al. paper are not from their strongest formulation, but from an enhanced single commodity ow formulation. A significant contribution of this paper is in developing a branch-and-cut formulation to provide improved lower bounds that can be computed in a reasonable amount of time.
- Table 6 allows a direct comparison of the upper bounds found by the original GRASP in [19] and by the new GRASP using the embedded branch-and-cut. Improvements were found in 49 out of the 50 instances in series 150c. The comparison in this case is also fair, since running times are similar. It can also be noted that a single iteration of the new GRASP is usually better than 10 iterations of the original one.
- As an additional experiment not shown in those tables, we also ran the new GRASP over series 100c. In average, the first iteration finds upper bounds that are 0.14% above the optimal (15 optimal solutions) in 386 seconds. The GRASP with 10 iterations finds upper bounds that are 0.005% above the optimal (47 optimal solutions).

References

[1] R.K. Ahuja, J.B. Orlin, and D. Sharma, "Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem", *Mathematical Programming* 91 (2001), 71 – 97.

[2] R.K. Ahuja, J.B. Orlin, and D. Sharma, "A composite very large-scale neighborhood structure for the capacitated minimum spanning tree problem", *Operations Research Letters* 31 (2003), 185 – 194.

[3] A. Amberg, W. Domschke, and S. Voss, "Capacitated minimum spanning tree: Algorithms using intelligent search", *Combinatorial Optimization: Theory and Practice* 1 (1996), 9 – 40.

[4] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "TSP cuts which do not conform to the template paradigm", *Computational Combinatorial Optimization*, Springer, (2001), 261–303.

[5] P. Avella, M. Boccia, and I. Vasilyev, "A computational study of exact knapsack separation for the generalized assignment problem", *Computational Optimization and Applications* 45 (2010), 543-555.

[6] E.A. Boyd, "Generating Fenchel cutting planes for knapsack polyhedra", *SIAM Journal on optimization* 3 (1993), 734–750.

[7] E.A. Boyd, "Fenchel cutting planes for integer programs", *Operations Research* 41 (1994), 53–64.

[8] S. Dash, R. Fukasawa and O. Günlük, "On a generalization of the master cyclic group polyhedron", *Mathematical Programming*, on-line first (2010).

[9] L.R. Esau and K.C. Williams, "On teleprocessing system design", *IBM Systems Journal* 5 (1966), 142 – 147.

[10] P. Festa and M.G.C. Resende, "An annotated bibliography of GRASP–Part II: Applications", *International Transactions in Operational Research* 16 (2009), 131–172.

[11] R. Fukasawa and M. Goycoolea, "On the exact separation of mixed-integer knapsack cuts", *Mathematical Programming*, on-line first (2010).

[12] I. Gamvros, S. Raghavan, and B.L. Golden, "An evolutionary approach for the multi-level capacitated minimum spanning tree", **In:** *Telecommunications Network Design and Management*, G. Anandalingam and S. Raghavan (Editors), Kluwer Academic Publishers, 2003, 99 – 124.

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

[14] L. Gouveia, "A $2n$ formulation for the capacitated minimal spanning tree problem", *Operations Research* 43 (1995), 130 – 141.

[15] L. Gouveia and P. Martins, "A hierarchy of hop-indexed models for the capacitated minimum spanning tree problem", *Networks* 35 (2000), 1 – 16.

[16] L. Gouveia and P. Martins, "The capacitated minimum spanning tree problem: revisiting hop-indexed formulations", *Computers & Operations Research* 32 (2005), 2435 – 2452.

[17] L. Hall, "Experience with a cutting plane approach for the capacitated spanning tree problem", *INFORMS Journal on Computing* 8 (1996), 219 – 234.

[18] P. Martins, "Enhanced second order algorithm applied to the capacitated minimum spanning tree problem", *Computers & Operations Research* 34 (2007), 2495 – 2519.

[19] 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* 15 (2009), 133 – 151.

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

[21] M. Ramos, "Solving capacitated facility location problems by Fenchel cutting planes", *Journal of the Operational Research Society* 56 (2005), 297–306.

[22] H. Santos, E. Uchoa, L. Ochi, and N. Maculan, "Strong bounds with cut and column generation for class-teacher timetabling", *Annals of Operations Research* online-first (2010).

	LP	+HECC Heu.	+HECC Ex.	+Fenchel	
series	av. g(%)	av. g(%)	av. g(%)	av. g(%)	max g(%)
50c	6.50	0.67	0.26	0.15	0.45
50e	4.22	0.87	0.63	0.34	0.66
50r	5.18	0.81	0.51	0.27	0.63
100c	5.25	0.94	0.67	0.38	0.67

Table 1: Influence of each family of cuts in the root gaps (with respect to the optimal solutions).

[23] Y. Sharaiha, M. Gendreau, G. Laporte, and I. Osman, "A tabu search algorithm for the capacitated shortest spanning tree problem", *Networks* 29 (1997), 161 – 171.

[24] M.C. de Souza, C. Duhamel, C.C. Ribeiro, "A GRASP heuristic for the capacitated minimum spanning tree problem using a memory-based local search strategy", *Metaheuristics: Computer Decision-Making*, M.G.C. Resende and J.P. De Sousa (Editors), Kluwer Academic Publishers, 627 – 657, 2003.

[25] 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), 443 – 472.

[26] S. Voss, "Capacitated minimum spanning trees", *Encyclopedia of Optimization*, Vol. 1, C.A. Floudas and P.M. Pardalos (Editors), Springer, 225 – 235, 2008.

50c	OPT	time
0	568.476	16.6
1	540.621	17.9
2	558.659	15.0
3	564.283	24.0
4	541.677	14.6
5	608.158	17.1
6	571.427	21.3
7	580.527	18.3
8	616.956	13.2
9	635.477	18.7
10	557.311	17.6
11	592.567	19.9
12	630.495	18.2
13	541.065	20.3
14	565.96	15.8
15	560.875	18.3
16	577.772	11.2
17	570.136	18.8
18	605.112	11.4
19	608.206	19.1
20	561.301	27.5
21	593.014	14.7
22	565.352	19.2
23	573.494	24.9
24	623.627	11.6
25	554.543	3.6
26	570.778	14.4
27	564.583	21.0
28	581.974	12.3
29	572.327	19.5
30	601.977	15.3
31	582.82	20.4
32	591.413	24.7
33	562.659	11.8
34	596.988	15.3
35	574.187	20.3
36	579.59	18.6
37	525.88	12.3
38	595.329	17.2
39	549.045	9.5
40	554.145	11.0
41	578.868	21.7
42	581.187	23.2
43	581.138	12.5
44	584.498	21.6
45	581.792	25.6
46	575.617	16.1
47	622.094	21.8
48	579.057	19.5
49	559.609	7.7
Avg.		17.2

50e	OPT	time
0	1108.674	47.6
1	1147.728	153.5
2	1007.268	55.9
3	1084.108	115.2
4	1123.234	129.8
5	1096.209	138.4
6	1002.304	54.3
7	1038.709	65.3
8	1077.996	26.5
9	1117.051	165.2
10	1115.708	108.8
11	1093.425	146.3
12	1018.228	143.6
13	1158.830	58.0
14	1093.222	37.1
15	1081.795	59.7
16	1044.068	121.4
17	1058.846	183.9
18	1073.215	169.7
19	1039.356	102.8
20	1042.514	28.9
21	1138.702	112.7
22	1107.963	282.7
23	1020.009	102.7
24	1064.897	41.1
25	1030.405	116.9
26	1121.626	31.7
27	941.540	76.6
28	1059.916	461.3
29	1128.082	47.5
30	1044.712	81.3
31	1109.926	126.7
32	1135.531	69.4
33	1068.535	79.0
34	1024.628	83.2
35	1039.296	41.1
36	1057.711	59.4
37	1001.467	188.9
38	1080.916	175.5
39	1038.405	83.2
40	1046.796	155.1
41	1049.914	84.3
42	1052.056	137.4
43	1048.190	70.1
44	1017.236	115.6
45	1061.875	134.4
46	1106.692	157.3
47	1113.141	83.9
48	1028.043	191.6
49	1098.593	158.0
Avg.		113.2

Table 2: Branch-and-cut times and optimal values: series 50c and 50e.

50r	OPT	time
0	591.991	19.0
1	737.046	48.1
2	701.633	22.4
3	676.355	20.2
4	859.786	49.4
5	958.692	278.5
6	767.158	43.2
7	741.734	76.0
8	684.667	45.4
9	829.176	38.1
10	850.330	42.7
11	670.775	16.0
12	715.212	32.2
13	790.461	21.5
14	786.882	26.0
15	872.357	108.9
16	702.672	16.2
17	585.497	18.6
18	690.706	18.1
19	811.374	143.4
20	879.342	48.2
21	797.794	64.9
22	688.662	24.1
23	720.743	80.8
24	809.756	93.7
25	625.111	19.6
26	853.640	114.8
27	694.711	30.3
28	773.762	41.8
29	1062.233	85.8
30	712.625	34.1
31	729.420	31.0
32	973.967	109.6
33	970.959	62.0
34	792.202	26.6
35	551.251	21.6
36	698.521	22.3
37	769.785	38.1
38	657.630	22.6
39	687.322	42.5
40	649.140	24.0
41	788.533	28.4
42	535.245	20.5
43	719.962	16.1
44	621.563	18.7
45	800.283	45.0
46	547.649	22.2
47	766.440	31.0
48	684.083	19.4
49	628.967	13.1
Avg.		46.7

100c	OPT	time
0	1075.429	219.5
1	1102.352	557.5
2	1108.356	3372.3
3	1096.077	293.8
4	1073.83	1886.8
5	1106.464	381.9
6	1042.484	14009.2
7	1136.44	9411.8
8	1104.339	2895.5
9	1140.361	9924.5
10	1040.935	1242.6
11	1046.843	241.1
12	1100.986	2310.2
13	1076.326	883.7
14	1052.864	3704.7
15	1096.602	9677.5
16	1081.679	713.3
17	1196.528	41088.1
18	1100.988	242.7
19	1064.041	194.3
20	1102.979	1014.5
21	1082.299	3981.2
22	1026.593	448.6
23	1088.228	1009.0
24	1095.762	3577.1
25	1088.681	1658.3
26	1058.178	4125.2
27	1067.799	2894.5
28	1071.537	39594.0
29	1136.181	1057.7
30	1036.521	4381.4
31	1062.344	304.0
32	1059.192	113694.8
33	1105.185	689.7
34	1071.753	3282.1
35	1128.925	873.0
36	1123.295	3554.4
37	1114.458	1491.8
38	1123.318	1495.8
39	1095.434	414.5
40	1087.756	16510.1
41	1070.239	10466.4
42	1004.472	1715.8
43	1059.29	1790.1
44	1098.467	1632.9
45	1135.27	1080.5
46	1120.805	5932.0
47	1132.807	1526.1
48	1061.246	523.4
49	1034.512	5035.4
Avg.		6780.1

Table 3: Branch-and-cut times and optimal values: series 50r and 100c.

	Best Known [13, 20]			GRASP + B&C						
100e	UB	LB	g(%)	UB1	UB10	time1	time10	LB	time	g(%)
0	2185.18	2102.98	3.91	2170.79	2168.097	855.6	6413.7	2158.87	532.8	0.43
1	2015.96	1943.44	3.73	2018.45	2007.625	709.2	5895.8	2000.41	1392.2	0.36
2	2120.68	2054.84	3.20	2113.19	2111.526	533.4	5258.1	2103.56	326.1	0.38
3	1993.73	1928.66	3.37	1991.74	1989.065	677.8	5827.2	1978.62	600.6	0.53
4	2037.46	1966.56	3.61	2038.51	2032.877	700.3	5916.6	2020.18	1136.3	0.63
5	2150.73	2073.86	3.71	2139.10	2136.933	341.5	4872.8	2127.46	665.0	0.45
6	2037.07	1971.26	3.34	2029.86	2028.611	759.5	6843.9	2019.39	706.0	0.46
7	2222.60	2148.97	3.43	2210.14	2209.246	576.4	5276.8	2201.38	498.8	0.36
8	2044.59	1963.01	4.16	2029.32	2029.322	811.6	7058.7	2018.46	463.2	0.54
9	2033.73	1961.01	3.71	2028.30	2021.453	460.7	4820.0	2012.16	563.8	0.46
10	2016.62	1941.98	3.84	2001.89	2000.862	764.5	5734.2	1992.15	352.6	0.44
11	2222.48	2146.83	3.52	2208.08	2208.082	331.8	5261.5	2201.86	505.2	0.28
12	1924.49	1852.91	3.86	1926.17	1921.888	436.6	5927.4	1910.28	441.6	0.61
13	2145.84	2066.34	3.85	2137.65	2136.541	645.1	6200.5	2126.83	735.7	0.46
14	2061.59	1982.66	3.98	2049.28	2049.284	450.1	6328.6	2040.32	391.7	0.44
15	1965.80	1901.06	3.41	1958.92	1958.919	782.4	5433.2	1950.70	489.4	0.42
16	2095.43	2030.00	3.22	2088.47	2088.468	1132.8	12556.1	2079.36	571.4	0.44
17	1967.80	1891.82	4.02	1961.38	1957.885	1738.7	19534.3	1946.05	442.7	0.61
18	2051.35	1981.84	3.51	2049.54	2045.390	1231.8	17404.5	2034.88	409.5	0.52
19	2194.22	2114.75	3.76	2182.52	2175.994	1700.7	18579.3	2168.12	861.8	0.36
20	1941.27	1869.10	3.86	1933.86	1932.486	2050.1	20092.4	1923.29	865.8	0.48
21	2137.95	2068.23	3.37	2129.83	2126.715	1778.8	16530.3	2118.20	478.6	0.40
22	2021.83	1944.19	3.99	2018.44	2013.261	1578.3	20806.0	2001.92	580.1	0.57
23	2055.94	1980.08	3.83	2045.04	2045.042	1189.1	9713.3	2036.78	595.4	0.41
24	2029.21	1948.85	4.12	2023.51	2018.398	1301.5	11418.0	2007.50	510.1	0.54
25	1954.69	1882.28	3.85	1948.78	1941.825	795.6	10431.5	1932.16	496.4	0.50
26	2108.91	2029.42	3.92	2098.98	2091.216	926.6	9457.5	2084.55	649.8	0.32
27	1978.95	1900.46	4.13	1967.51	1966.986	1473.5	12655.5	1953.46	366.2	0.69
28	2105.44	2022.93	4.08	2099.57	2092.897	1553.2	12167.9	2084.57	487.0	0.40
29	2104.96	2031.88	3.60	2105.13	2100.967	1065.0	12533.6	2086.02	443.7	0.72
30	1961.14	1882.63	4.17	1953.92	1947.711	1441.1	11988.0	1939.17	1026.1	0.44
31	2027.19	1957.09	3.58	2017.30	2012.820	580.4	8135.7	2005.88	278.0	0.35
32	2189.84	2116.81	3.45	2186.94	2174.858	990.1	10105.5	2167.66	365.3	0.33
33	2136.06	2055.51	3.92	2128.21	2126.354	1158.8	11097.3	2113.95	679.3	0.59
34	2072.05	1995.73	3.82	2064.52	2057.769	1190.6	9440.7	2050.44	806.5	0.36
35	2169.38	2092.02	3.70	2164.47	2159.746	1424.7	11999.3	2150.89	804.1	0.41
36	2009.65	1934.03	3.91	1996.45	1995.119	1174.2	11622.5	1986.77	425.4	0.42
37	2037.62	1953.39	4.31	2019.98	2018.325	813.4	8321.0	2012.11	426.9	0.31
38	2136.86	2059.57	3.75	2125.95	2120.883	1208.0	11518.8	2110.61	363.1	0.49
39	2037.17	1975.43	3.13	2036.30	2031.733	762.6	9797.5	2025.61	952.0	0.30
40	2037.46	1972.76	3.28	2037.98	2030.022	1016.3	9956.2	2021.67	521.1	0.41
41	2062.96	1991.65	3.58	2059.63	2048.908	1001.1	9641.6	2043.58	614.2	0.26
42	1884.68	1811.11	4.06	1878.88	1875.663	952.5	11367.5	1866.39	424.8	0.50
43	2233.21	2155.90	3.59	2227.63	2222.880	983.0	11645.5	2211.72	494.4	0.50
44	2164.30	2093.22	3.40	2158.70	2152.697	821.3	9674.0	2146.49	587.5	0.29
45	2170.40	2091.25	3.79	2155.22	2155.208	1437.8	12203.5	2145.81	482.3	0.44
46	2138.39	2058.52	3.88	2129.36	2127.234	799.0	12224.9	2115.79	685.6	0.54
47	2060.37	1987.68	3.66	2052.92	2047.853	981.8	9124.6	2041.88	582.9	0.29
48	2052.43	1982.35	3.54	2046.09	2041.061	1143.6	7583.2	2031.39	298.4	0.48
49	2067.58	1991.97	3.80	2055.66	2054.915	534.4	5046.7	2042.68	322.7	0.60
Avg.	2071.50	1997.21	3.72	2064.00	2060.192	995.3	9988.8	2051.00	574.0	0.45

Table 4: Results for the 100e series.

	Best Known [13, 20]			GRASP + B&C						
100r	UB	LB	g(%)	UB1	UB10	time1	time10	LB	time	g(%)
0	1606.57	1533.735	4.75	1594.36	1594.358	363.1	5953.6	1588.09	438.1	0.39
1	1893.58	1816.662	4.23	1885.38	1885.063	653.5	6571.5	1875.21	489.5	0.53
2	1488.34	1418.726	4.91	1482.11	1476.980	455.5	5196.0	1473.08	276.6	0.26
3	1175.05	1108.194	6.03	1168.11	1166.972	433.7	3807.5	1159.71	418.3	0.63
4	1166.97	1101.724	5.92	1158.40	1155.714	396.0	3347.2	1149.27	187.7	0.56
5	1216.15	1148.867	5.86	1207.35	1202.533	235.7	3796.2	1197.87	178.5	0.39
6	1612.01	1543.942	4.41	1603.82	1602.116	797.2	5430.7	1594.86	302.9	0.45
7	1116.79	1048.669	6.50	1109.79	1107.746	421.8	3814.7	1100.72	627.7	0.64
8	1342.13	1266.527	5.97	1336.49	1326.983	786.5	4678.8	1321.46	720.0	0.42
9	1511.44	1431.159	5.61	1497.35	1494.310	462.3	5123.9	1484.94	286.5	0.63
10	1501.27	1424.499	5.39	1493.59	1490.672	685.9	6368.5	1479.14	459.2	0.78
11	1800.73	1724.491	4.42	1797.68	1793.842	407.2	6152.4	1785.39	640.1	0.47
12	1713.92	1642.249	4.36	1708.38	1706.713	612.7	5498.6	1695.12	530.3	0.68
13	1385.85	1311.500	5.67	1369.53	1369.532	628.0	4693.7	1364.20	484.2	0.39
14	1260.43	1187.978	6.10	1245.24	1245.238	308.8	2950.6	1239.54	144.6	0.46
15	1621.79	1551.197	4.55	1616.86	1616.831	430.3	4901.9	1607.43	753.7	0.59
16	1142.20	1077.067	6.05	1132.96	1132.185	449.3	4075.5	1128.42	250.9	0.33
17	1327.48	1251.539	6.07	1321.87	1315.928	445.3	4314.2	1307.93	224.3	0.61
18	1730.69	1661.108	4.19	1726.67	1724.015	428.8	4879.5	1717.51	472.6	0.38
19	1122.80	1060.098	5.91	1117.57	1111.798	313.3	3159.7	1109.53	315.2	0.20
20	1580.73	1508.644	4.78	1572.33	1570.814	432.9	4095.9	1561.07	348.0	0.62
21	1515.29	1442.315	5.06	1511.85	1501.877	417.8	4536.9	1494.59	297.4	0.49
22	1155.67	1090.206	6.00	1140.39	1140.389	191.3	2403.9	1138.96	325.1	0.13
23	1250.89	1178.639	6.13	1236.36	1232.631	269.3	3491.2	1227.87	239.7	0.39
24	1510.58	1438.406	5.02	1505.15	1505.148	572.2	4870.3	1496.28	335.6	0.59
25	1612.11	1541.554	4.58	1595.20	1595.197	321.2	4424.0	1589.81	509.2	0.34
26	1376.62	1312.655	4.87	1370.55	1369.210	320.4	3436.4	1361.33	223.8	0.58
27	1546.52	1470.298	5.18	1535.43	1535.426	430.7	4552.4	1526.82	1367.3	0.56
28	1128.19	1059.286	6.50	1123.84	1116.913	385.3	3515.3	1109.76	213.6	0.64
29	1217.10	1149.665	5.87	1211.62	1210.074	441.5	3505.1	1203.83	289.5	0.52
30	1278.78	1203.966	6.21	1264.27	1264.267	310.2	3054.5	1258.52	456.6	0.46
31	1497.23	1422.378	5.26	1484.86	1484.862	496.3	4099.3	1477.33	337.6	0.51
32	1448.38	1377.455	5.15	1438.82	1436.271	296.1	3600.6	1430.81	609.5	0.38
33	1601.11	1525.623	4.95	1598.76	1593.734	364.3	4643.9	1581.10	244.8	0.80
34	1535.34	1460.583	5.12	1524.00	1521.744	944.8	4809.3	1513.13	740.7	0.57
35	1790.70	1708.371	4.82	1781.84	1777.035	367.9	5590.9	1767.97	635.4	0.51
36	1198.81	1127.950	6.28	1191.33	1190.458	360.3	3660.4	1183.35	458.8	0.60
37	1628.19	1556.451	4.61	1624.94	1622.724	598.4	6207.4	1612.90	471.8	0.61
38	1692.43	1607.992	5.25	1681.93	1674.090	516.1	5574.5	1665.72	839.3	0.50
39	1330.64	1253.470	6.16	1314.16	1311.144	530.0	4271.1	1305.48	619.2	0.43
40	1514.00	1441.186	5.05	1501.32	1500.573	431.7	3686.4	1494.12	387.5	0.43
41	1205.97	1135.890	6.17	1192.53	1192.525	295.3	3272.1	1187.92	188.2	0.39
42	1813.27	1736.099	4.45	1808.89	1804.179	999.5	6397.0	1792.77	399.1	0.64
43	1576.17	1501.478	4.97	1568.29	1562.929	402.9	4814.6	1555.75	317.4	0.46
44	1760.82	1684.780	4.51	1751.65	1746.806	466.9	5156.1	1739.34	728.2	0.43
45	1499.78	1429.201	4.94	1501.93	1496.918	529.3	4987.1	1486.65	221.9	0.69
46	1164.72	1095.817	6.29	1154.47	1154.473	279.7	3847.7	1146.19	147.4	0.72
47	1411.14	1348.263	4.66	1405.88	1405.875	544.8	4157.8	1397.64	191.5	0.59
48	1993.48	1906.716	4.55	1985.71	1977.807	596.2	6265.4	1966.93	603.3	0.55
49	1710.07	1631.454	4.82	1696.35	1696.345	455.5	4407.9	1687.07	541.8	0.55
Avg.	1465.62	1393.13	5.30	1456.96	1454.239	465.7	4521.0	1446.81	429.8	0.51

Table 5: Results for the 100r series.

	Best Known			GRASP + B&C						
150c	UB[19]	LB[20]	g(%)	UB1	UB10	time1	time10	LB	time	g(%)
0	1555.09	1483.97	4.79	1551.20	1550.382	695.6	7614.3	1541.393	422.7	0.58
1	1639.31	1569.58	4.44	1638.97	1634.420	560.5	5587.7	1627.047	569.2	0.45
2	1624.75	1550.35	4.80	1624.90	1620.860	675.1	5680.6	1611.212	673.6	0.60
3	1586.54	1509.93	5.07	1578.88	1578.877	551.5	5996.6	1569.488	588.9	0.60
4	1633.25	1555.02	5.03	1630.15	1626.312	559.8	5594.2	1617.757	990.1	0.53
5	1658.82	1583.08	4.78	1656.97	1655.029	566.4	5550.8	1643.616	617.9	0.69
6	1560.93	1486.18	5.03	1553.34	1549.002	528.0	5436.9	1542.732	415.8	0.41
7	1601.46	1520.71	5.31	1595.32	1593.127	693.6	7167.6	1583.200	456.8	0.63
8	1584.62	1513.96	4.67	1586.31	1584.944	470.7	6203.3	1576.648	477.1	0.53
9	1551.76	1477.30	5.04	1548.26	1548.265	680.2	6689.5	1536.034	446.1	0.80
10	1670.66	1589.69	5.09	1670.10	1664.195	695.6	6760.5	1654.867	481.0	0.56
11	1536.38	1468.01	4.66	1542.59	1535.340	767.9	7527.4	1527.398	438.5	0.52
12	1600.25	1526.61	4.82	1596.21	1596.207	651.5	7783.8	1587.526	426.8	0.55
13	1620.42	1548.12	4.67	1619.83	1617.850	389.2	6168.8	1609.479	720.8	0.52
14	1596.45	1520.68	4.98	1592.23	1586.822	523.6	6798.2	1579.459	651.8	0.47
15	1591.07	1522.01	4.54	1588.50	1588.500	563.5	6328.7	1577.515	484.0	0.70
16	1546.31	1466.95	5.41	1543.16	1540.703	777.0	7755.3	1530.685	636.7	0.65
17	1642.19	1566.31	4.84	1640.83	1638.974	786.5	8364.9	1630.967	698.8	0.49
18	1627.52	1548.84	5.08	1629.76	1621.329	594.8	7203.0	1614.072	709.6	0.45
19	1674.52	1600.14	4.65	1677.15	1673.395	717.3	7239.8	1663.916	374.9	0.57
20	1524.37	1452.73	4.93	1520.09	1520.092	465.6	6895.5	1512.048	415.2	0.53
21	1614.36	1542.23	4.68	1613.01	1609.188	613.7	7275.4	1600.282	594.8	0.56
22	1615.52	1539.50	4.94	1610.55	1607.782	583.4	7132.5	1600.146	537.7	0.48
23	1611.31	1538.72	4.72	1609.58	1606.530	686.7	7326.1	1595.303	517.6	0.70
24	1626.81	1552.27	4.80	1625.02	1624.085	624.3	7683.4	1614.448	1003.8	0.60
25	1588.24	1512.61	5.00	1585.48	1582.446	709.8	6550.4	1576.052	934.4	0.41
26	1564.28	1485.25	5.32	1556.01	1556.013	923.2	8393.3	1548.540	311.0	0.48
27	1660.21	1581.59	4.97	1661.34	1655.092	761.7	6689.3	1644.906	664.7	0.62
28	1573.74	1506.52	4.46	1570.06	1568.964	682.8	6546.9	1560.714	451.6	0.53
29	1580.15	1502.69	5.15	1576.58	1573.068	649.7	6530.8	1563.377	445.0	0.62
30	1585.97	1511.77	4.91	1593.67	1582.869	731.6	7394.2	1573.786	517.1	0.58
31	1608.11	1529.41	5.15	1604.06	1598.005	746.3	8243.3	1590.344	1025.4	0.48
32	1599.57	1519.52	5.27	1597.22	1597.216	730.3	7917.5	1583.550	492.2	0.86
33	1646.67	1572.18	4.74	1645.09	1642.690	773.0	6696.1	1632.541	731.0	0.62
34	1523.06	1451.05	4.96	1519.33	1519.330	760.5	7018.1	1512.086	432.2	0.48
35	1626.49	1548.54	5.03	1626.39	1624.873	822.8	7894.1	1612.816	638.6	0.75
36	1525.58	1456.26	4.76	1530.04	1524.946	681.5	6196.5	1513.877	295.9	0.73
37	1576.40	1509.76	4.41	1575.98	1573.387	907.4	7448.7	1564.809	600.9	0.55
38	1519.99	1441.42	5.45	1523.14	1513.890	603.5	7896.7	1504.379	477.8	0.63
39	1584.22	1520.04	4.22	1584.44	1583.614	670.2	6102.7	1575.787	488.8	0.50
40	1673.15	1595.44	4.87	1670.16	1668.653	520.5	6555.5	1660.890	755.8	0.47
41	1608.74	1530.98	5.08	1609.97	1605.475	891.8	8274.7	1592.841	365.6	0.79
42	1626.90	1554.19	4.68	1626.37	1622.238	563.1	6072.8	1614.284	518.0	0.49
43	1610.01	1539.88	4.55	1607.64	1604.058	884.4	7605.3	1594.092	977.5	0.63
44	1613.59	1546.49	4.34	1614.48	1609.328	381.7	6563.5	1604.367	520.0	0.31
45	1630.23	1554.09	4.90	1631.16	1626.152	714.2	6754.1	1617.772	518.1	0.52
46	1641.71	1567.32	4.75	1638.78	1634.553	428.4	5375.6	1628.407	1203.4	0.38
47	1610.86	1534.33	4.99	1603.05	1600.638	739.9	6357.3	1593.734	508.9	0.43
48	1597.07	1523.08	4.86	1596.67	1592.144	631.9	8128.3	1582.404	651.3	0.62
49	1668.59	1588.24	5.06	1662.51	1658.807	760.9	7804.2	1651.313	822.7	0.45
Avg.	1602.76	1528.31	4.87	1601,05	1597.813	661.9	6935.5	1588.90	594.0	0.56

Table 6: Results for the 150c series.

	Best Known [13, 20]			GRASP + B&C						
150e	UB	LB	g(%)	UB1	UB10	time1	time10	LB	time	g(%)
0	3043.19	2932.04	3.79	3003.72	3003.718	710.8	8678.9	2992.415	728.1	0.38
1	3130.60	3017.36	3.75	3102.86	3099.175	829.2	9653.3	3082.438	1004.1	0.54
2	3074.82	2960.41	3.86	3049.30	3043.186	754.2	9466.2	3028.971	1086.1	0.47
3	3049.50	2944.37	3.57	3028.94	3025.921	905.5	9004.7	3008.361	1129.9	0.58
4	3057.15	2930.99	4.30	3013.03	3012.149	734.6	8933.0	2993.525	976.8	0.62
5	3124.30	3018.28	3.51	3105.76	3096.325	1065.6	9152.7	3080.425	1460.3	0.52
6	3139.49	3020.23	3.95	3104.05	3104.051	1246.1	10648.3	3083.835	769.8	0.66
7	3124.99	3018.03	3.54	3102.89	3101.611	949.6	8463.1	3080.592	684.1	0.68
8	3027.51	2899.28	4.42	2993.71	2987.788	1112.1	9760.5	2968.513	1082.6	0.65
9	3033.71	2916.53	4.02	2998.24	2998.236	1119.8	10077.9	2981.108	848.8	0.57
10	3085.52	2967.32	3.98	3050.02	3044.015	1578.5	17067.4	3029.875	732.5	0.47
11	3061.33	2959.22	3.45	3039.20	3038.636	1650.4	15949.6	3021.685	818.3	0.56
12	3052.91	2930.67	4.17	3022.82	3014.345	1727.9	15944.5	2999.968	1487.2	0.48
13	3111.28	3000.72	3.68	3085.13	3083.403	1466.8	17285.2	3067.367	1152.0	0.52
14	3107.94	2985.61	4.10	3076.49	3066.980	1749.1	17758.5	3052.041	1224.4	0.49
15	3092.81	2979.22	3.81	3065.49	3060.121	1723.4	18581.7	3045.106	971.5	0.49
16	2904.96	2799.28	3.78	2884.08	2876.751	940.6	10063.2	2860.175	630.6	0.58
17	3150.99	3030.04	3.99	3112.98	3111.366	1591.6	18793.6	3095.731	975.5	0.51
18	3078.75	2978.55	3.36	3065.20	3056.673	1508.1	15589.4	3043.994	971.9	0.42
19	2928.83	2815.16	4.04	2897.04	2896.769	1952.7	18490.1	2878.857	1089.8	0.62
20	3036.93	2921.85	3.94	3007.06	2992.853	2114.4	17352.8	2983.935	1018.7	0.30
21	3174.22	3060.85	3.70	3139.39	3139.386	1259.3	13981.1	3123.488	996.6	0.51
22	3105.00	3009.46	3.17	3089.37	3077.937	1458.3	14976.1	3064.579	1046.6	0.44
23	3005.69	2899.90	3.65	2984.77	2980.575	1615.5	17523.1	2962.699	1219.6	0.60
24	3098.07	2988.50	3.67	3071.71	3065.603	1259.0	15485.4	3048.968	675.3	0.55
25	3052.12	2937.04	3.92	3017.29	3011.713	1246.7	14651.7	2999.329	633.7	0.41
26	3038.44	2925.25	3.87	3005.36	3002.238	1917.6	18359.8	2987.528	830.8	0.49
27	2892.99	2781.39	4.01	2862.36	2861.963	1261.6	16174.6	2842.803	970.9	0.67
28	2974.59	2854.93	4.19	2944.11	2936.602	1230.2	15514.4	2921.027	1354.6	0.53
29	3151.54	3043.55	3.55	3136.00	3124.704	1766.7	17390.2	3106.190	1055.4	0.60
30	3127.57	3022.07	3.49	3104.21	3100.487	3327.4	20869.4	3084.961	1037.3	0.50
31	3037.74	2930.86	3.65	3004.89	3004.895	1476.7	17195.7	2989.889	958.2	0.50
32	3161.11	3044.55	3.83	3134.95	3127.715	1717.1	18160.5	3111.192	987.6	0.53
33	3195.45	3074.38	3.94	3161.80	3154.333	1586.1	18423.5	3141.674	905.0	0.40
34	3139.71	3024.30	3.82	3111.90	3103.323	2068.6	17934.9	3089.086	1045.0	0.46
35	3095.11	2980.82	3.83	3066.32	3059.124	2351.3	17065.1	3042.552	1303.9	0.54
36	2972.63	2862.10	3.86	2946.51	2942.259	1763.3	18329.3	2924.125	925.5	0.62
37	3000.31	2888.35	3.88	2970.54	2966.372	1773.4	19315.4	2951.453	803.6	0.51
38	3187.47	3079.74	3.50	3163.27	3160.703	1434.4	16549.4	3143.987	1317.9	0.53
39	2976.62	2850.34	4.43	2946.36	2939.031	1333.8	17531.8	2921.866	1017.4	0.59
40	3050.83	2941.69	3.71	3026.22	3021.389	2348.3	19884.4	3004.662	1119.8	0.56
41	2972.10	2867.58	3.64	2949.15	2948.040	2000.0	16282.6	2931.986	1154.0	0.55
42	3104.22	2979.04	4.20	3064.21	3057.400	1651.2	18780.0	3043.192	1116.2	0.47
43	2966.58	2871.71	3.30	2952.43	2947.803	1520.3	16894.1	2930.487	1287.8	0.59
44	3084.01	2970.05	3.84	3053.11	3051.160	1892.2	17883.2	3032.430	1301.6	0.62
45	3150.20	3032.98	3.86	3116.22	3110.412	1717.8	17028.7	3093.925	958.0	0.53
46	3024.37	2904.06	4.14	2994.11	2985.329	1678.6	16473.5	2969.127	918.9	0.55
47	3120.89	2999.22	4.06	3082.37	3072.505	1958.2	17113.4	3062.022	1146.1	0.34
48	3058.92	2955.44	3.50	3037.05	3036.614	1627.7	17350.4	3021.196	1855.6	0.51
49	3143.35	3024.59	3.93	3117.60	3109.285	1637.5	19325.9	3094.408	1092.9	0.48
Avg.	3069.59	2956.60	3.82	3041.23	3036.259	1546.2	15583.2	3020.395	1037.6	0.53

Table 7: Results for the 150e series.

	Best Known [13, 20]			GRASP + B&C						
150r	UB	LB	g(%)	UB1	UB10	time1	time10	LB	time	g(%)
0	2418.80	2303.13	5.02	2384.15	2374.978	1198.6	9565.4	2363.65	692.2	0.48
1	2122.23	2020.05	5.06	2094.01	2087.433	680.4	7762.3	2076.39	761.9	0.53
2	2243.70	2135.84	5.05	2217.36	2216.420	881.1	8107.5	2201.89	500.7	0.66
3	2180.74	2076.64	5.01	2157.34	2147.615	777.7	7594.7	2135.20	742.8	0.58
4	2236.04	2128.82	5.04	2206.69	2206.680	552.0	8410.6	2195.59	829.8	0.51
5	2644.18	2520.64	4.90	2605.67	2601.696	1007.9	8973.5	2584.25	876.0	0.68
6	2472.68	2365.40	4.54	2443.17	2438.750	1252.7	7632.2	2421.91	721.4	0.70
7	2049.93	1940.63	5.63	2018.55	2011.802	565.9	6234.7	2001.65	773.7	0.51
8	2493.97	2368.44	5.30	2454.91	2452.279	1244.3	10030.4	2433.12	649.7	0.79
9	2053.54	1947.00	5.47	2016.09	2011.395	713.5	6933.9	2003.48	498.2	0.40
10	2249.45	2144.27	4.91	2229.50	2228.520	707.4	7718.7	2211.55	601.5	0.77
11	1956.35	1854.25	5.51	1930.20	1930.196	648.3	5953.2	1915.00	537.2	0.79
12	2269.84	2160.57	5.06	2247.33	2235.641	548.1	6074.5	2225.44	583.1	0.46
13	1954.28	1851.00	5.58	1929.34	1922.582	817.5	5840.0	1910.26	540.5	0.64
14	2205.40	2120.34	4.01	2197.11	2194.228	628.8	6387.5	2179.88	968.8	0.66
15	2474.14	2375.30	4.16	2451.28	2446.956	811.5	7021.4	2435.18	594.6	0.48
16	2660.86	2559.14	3.97	2632.04	2628.578	624.5	7909.3	2616.42	675.2	0.46
17	2466.24	2363.34	4.35	2443.04	2439.881	481.3	6844.0	2426.15	707.2	0.57
18	1871.91	1773.09	5.57	1844.40	1841.278	569.6	6041.9	1831.74	571.9	0.52
19	2426.58	2312.68	4.92	2396.89	2388.308	594.6	7458.9	2374.09	613.2	0.60
20	2455.44	2352.09	4.39	2435.37	2422.387	851.8	6973.9	2410.77	963.5	0.48
21	2208.22	2093.63	5.47	2175.61	2172.824	419.4	5346.9	2157.83	594.1	0.69
22	1754.90	1653.44	6.14	1730.95	1727.429	420.3	5478.4	1714.26	872.5	0.77
23	1982.48	1871.19	5.95	1951.57	1946.446	383.1	5119.6	1937.45	636.6	0.46
24	2139.09	2039.28	4.89	2112.30	2110.409	474.9	6789.5	2098.18	605.6	0.58
25	1755.58	1661.22	5.68	1733.23	1726.731	797.9	6051.6	1720.32	554.2	0.37
26	2062.79	1966.17	4.91	2036.14	2036.135	549.5	5797.4	2025.09	650.8	0.55
27	2951.49	2844.54	3.76	2927.70	2919.079	855.3	7683.5	2907.12	851.6	0.41
28	1970.09	1870.52	5.32	1943.19	1940.450	675.1	6789.1	1927.81	699.8	0.66
29	1737.12	1648.19	5.40	1721.50	1714.470	655.6	5121.1	1704.72	784.7	0.57
30	1947.16	1852.30	5.12	1923.75	1915.543	577.3	5228.8	1907.97	692.2	0.40
31	2462.89	2352.11	4.71	2436.57	2421.065	740.3	9031.6	2410.93	563.7	0.42
32	1544.92	1447.03	6.77	1518.60	1516.882	587.7	6125.1	1506.01	699.6	0.72
33	2397.05	2291.02	4.63	2372.39	2367.190	698.2	8028.2	2354.13	851.1	0.55
34	2228.29	2111.71	5.52	2201.29	2193.709	529.3	8201.4	2178.98	1012.8	0.68
35	1723.08	1619.80	6.38	1687.83	1687.832	970.2	7501.0	1676.76	794.2	0.66
36	2148.73	2046.57	4.99	2124.19	2119.462	797.8	7518.7	2106.81	669.8	0.60
37	1942.58	1834.91	5.87	1912.23	1912.230	905.6	8150.7	1895.70	655.7	0.87
38	2245.47	2134.09	5.22	2215.22	2215.219	842.2	9300.5	2200.56	674.7	0.67
39	2269.26	2176.78	4.25	2248.13	2248.131	1030.3	9272.5	2232.64	566.8	0.69
40	2366.26	2260.44	4.68	2343.23	2333.694	1307.0	8637.5	2326.67	1120.5	0.30
41	2059.14	1958.05	5.16	2031.23	2028.016	536.3	6874.4	2018.60	580.2	0.47
42	2043.65	1929.05	5.94	2012.51	2000.458	679.6	7538.5	1991.31	605.0	0.46
43	2490.46	2385.12	4.42	2465.37	2462.882	904.6	8683.0	2447.85	664.7	0.61
44	1643.22	1560.68	5.29	1629.63	1626.556	774.4	6794.7	1615.49	669.2	0.69
45	2095.97	1981.78	5.76	2066.51	2062.106	896.4	8060.1	2045.66	714.3	0.80
46	1751.63	1652.41	6.00	1723.96	1722.407	859.5	6504.4	1712.07	376.5	0.60
47	2371.64	2263.58	4.77	2351.54	2345.285	848.9	9953.7	2331.92	1003.2	0.57
48	1855.69	1763.64	5.22	1831.70	1831.425	913.0	7792.3	1820.36	570.6	0.61
49	2217.28	2105.49	5.31	2179.55	2178.144	753.8	7799.1	2166.74	640.4	0.53
Avg.	2165.45	2060.95	5.14	2138.84	2134.196	750.8	7332.8	2121.871	695.6	0.58

Table 8: Results for the 150r series.