A GRASP heuristic for the capacitated minimum spanning tree problem using a memory-based local search strategy

Mauricio C. de Souza (mauricio@isima.fr)
LIMOS, Université Blaise Pascal, BP 125, 63173 Aubière Cedex, France

Christophe Duhamel (christophe.duhamel@isima.fr)
LIMOS, Université Blaise Pascal, BP 125, 63173 Aubière Cedex, France

Celso C. Ribeiro (celso@inf.puc-rio.br)
Department of Computer Science, Catholic University of Rio de Janeiro
Rua Marquês de São Vicente, 225, Rio de Janeiro, 22453-900, Brazil

October 30, 2002

Abstract. We describe a new neighborhood structure for the capacitated minimum spanning tree problem. This neighborhood structure is used by a local search strategy, leading to good trade-offs between solution quality and computation time. We also propose a GRASP with path-relinking heuristic. It uses a randomized version of a savings heuristic in the construction phase and an extension of the above local search strategy, incorporating some short term memory elements of tabu search. Computational results on benchmark problems illustrate the effectiveness of this approach, which is competitive with the best heuristics in the literature in terms of solution quality. The GRASP heuristic using a memory-based local search strategy improved the best known solution for some of the largest benchmark problem.

Keywords: Capacitated minimum spanning tree, metaheuristics, GRASP, local search, neighborhood reduction, short term memory, path-relinking

1. Introduction

Let $G = (V, E)$ be a connected undirected graph, where $V = \{0, 1, \ldots, n\}$ denotes the set of nodes and E is the set of edges. Non-negative integers c_e and b_i are associated respectively with each edge $e \in E$ and with each node $i \in V$. Given an integer Q and a special *central node* $r \in V$, the Capacitated Minimum Spanning Tree (CMST) problem consists of finding a minimum spanning tree T of G in terms of the edge costs, such that the sum of the node weights in each connected component of the graph induced in T by $V - \{r\}$ is less than or equal to Q.

The CMST problem is NP-hard (Papadimitriou, 1978) for $3 \leq Q \leq \lfloor |V|/2 \rfloor$ and has applications in the design of communication networks, see e.g. (Amberg et al., 1996; Gavish, 1982; Gouveia and Martins, 2000). Gouveia and Martins (1999) proposed a hop-indexed flow model which is a generalization of a single-commodity flow model proposed by Gavish (1983) and reviewed exact and lower bounding schemes, including earlier works of Gavish (Gavish, 1982; Gavish, 1983), the branch-and-bound algorithm of Malik and Yu

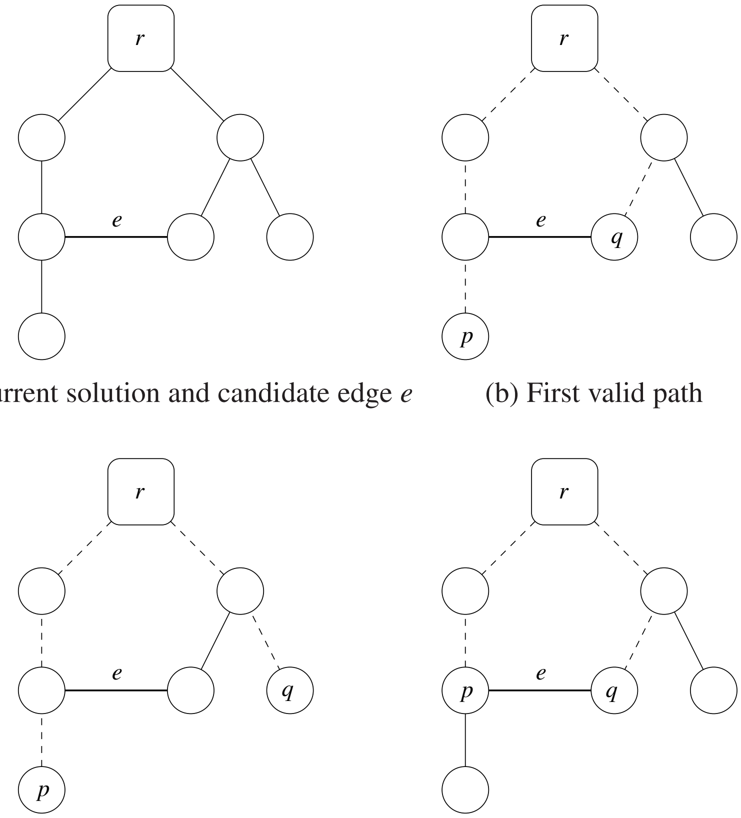
(1993), the Lagrangean relaxation approach of Gouveia (1995), and the cutting plane method of Hall (1996). Gouveia and Martins (2000) proposed an iterative method for computing lower bounds for the CMST problem, based on a hierarchy of hop-indexed linear programming models. Amberg et al. (1996) reviewed exact and approximate algorithms. Among the main heuristics, we find the savings procedure EW of Esau and Williams (1966) and the tabu search algorithms of Amberg et al. (1996) and Sharaiha et al. (1997). The neighborhood structure used in (Amberg et al., 1996) is based on exchanging single nodes between subtrees of the current solution. The neighborhood used in (Sharaiha et al., 1997) is an extension of the latter, in which parts of a subtree are moved from one subtree to another or to the central node. More recently, Ahuja et al. (2001) proposed new neighborhoods based on the cyclic exchange neighborhood described in (Thompson and Orlin, 1989; Thompson and Psaraftis, 1993) and developed GRASP and tabu search heuristics based on the concept of improvement graphs.

In this work, we propose a new GRASP with path-relinking heuristic for the capacitated minimum spanning tree problem. This heuristic uses a new local search strategy based on a different neighborhood structure defined by path exchanges, as described in the next section. Numerical results on benchmark problems are reported in Section 3, showing that the proposed local search strategy leads to good trade-offs between solution quality and computation time. The GRASP with path-relinking heuristic using a memory-based local search strategy is described in Section 4. Further computational results are reported in Section 5, illustrating the effectiveness of the heuristic. Concluding remarks and extensions are discussed in the last section.

2. Local search with a path-based neighborhood

Feasible initial solutions are constructed by the savings heuristic EW of Esau and Williams (1966). Starting from the r-centered star, this procedure joins the two components which yield maximum savings with respect to the edge costs and capacity constraints. The process iterates until no further savings can be achieved. It runs in $O(|V|^2 \log |V|)$ time.

Let T be a spanning tree of G, satisfying the capacity constraints. We define the components $T_i, i = 1, \dots, q$ of T as the subtrees induced in T by $V - \{r\}$. Solutions in the neighborhood of T are obtained as follows. Every edge $e \notin T$ with extremities in two different components T_i and T_j is considered as a candidate for insertion in the current tree. To speed up the search, we consider only a subset of candidate edges and we investigate the insertion of each of them. For each edge $e \in E$ considered for insertion, different forests can be obtained by removing each of at most Q^2 candidate paths with extremities in T_i and T_j. A candidate path may be removed if the tree containing



(a) Current solution and candidate edge e (b) First valid path

(c) Second valid path (d) Third valid path

Figure 1. The three valid paths for removal ($Q = 3$)

edge e in the resulting forest does not violate the capacity constraint. The example in Figure 1 (a) shows a feasible solution for a CMST instance with $Q = 3$ and a candidate edge e for insertion. Figures 1 (b) to (d) show the only three (out of nine) paths that may be removed without violating the capacity constraints in the resulting forest. Instead of evaluating all paths satisfying the above condition, we use an estimation of the reconnection costs to select a unique path for each candidate edge. Each component of the resulting forest is reconnected to the central node by the least cost edge and the savings procedure EW is applied from that solution. The structure of each move is sumarized in Figure 2. The best improving neighbor generated by the above scheme is chosen and the local search resumes from the new solution.

Each solution has $O(|E|Q^2)$ neighbors. For each edge $e \in E$ considered for insertion and for each path considered for removal, we evaluate in time $O(Q)$ whether this move is acceptable or not, by examining the nodes

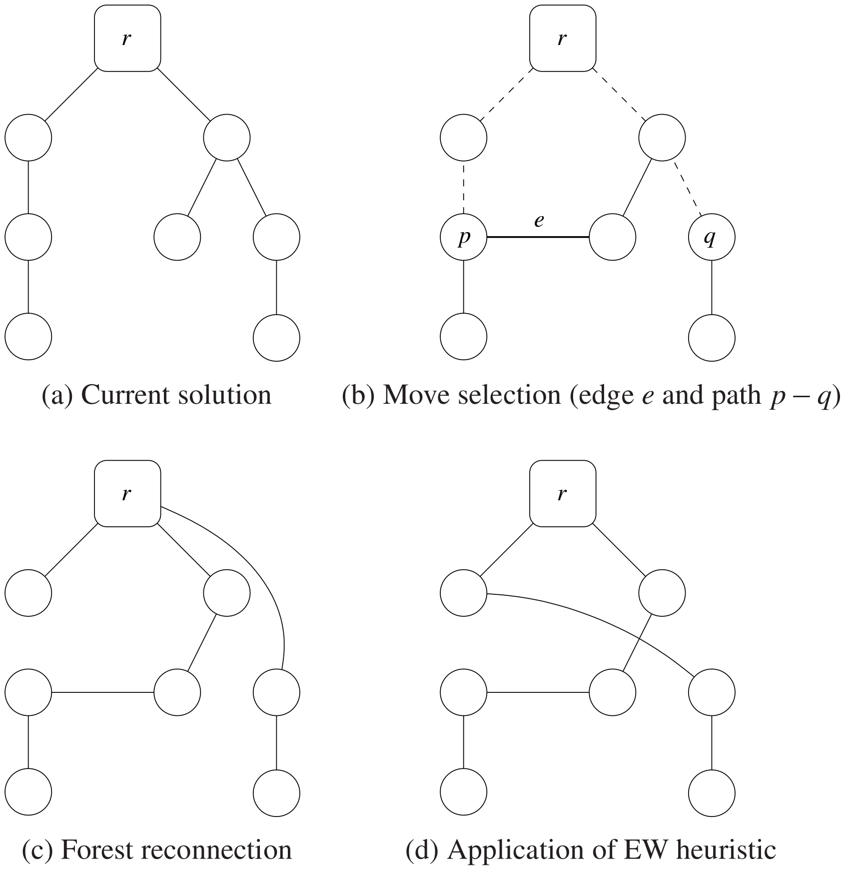


Figure 2. The four steps of a move

within the tree containing edge e. Since the reconnection of each acceptable move is computed by the savings procedure in $O(|V|^2 \log|V|)$ time, the complexity of the investigation of the neighborhood of each solution is $O(|E||V|^2Q^2 \log|V|)$, with $Q \leq \lfloor |V|/2 \rfloor$.

2.1. Selection of Candidate Edges

The neighborhood size has a strong influence on the computational performance of local search procedures. We use some heuristic selection rules to speed up the search by restricting the neighbor solutions investigated.

Each local search iteration starts by a depth-first search traversal of the current solution. For each node $i \in V - \{r\}$, we obtain its predecessor j in the incoming path from the central node and the capacity of the subtree rooted at i (i.e., the sum of the weights of all nodes in this subtree) if edge (i, j) is eliminated from the current tree.

Each move is based on the selection of an edge $e \in E$, followed by that of a path $(p-q)$ between two different components of the current solution. We propose three tests to reduce the number of moves evaluated. The first of them makes use of the relative costs associated with the edges:

DEFINITION 1 (Relative cost). *The relative cost r_e of an edge $e \in E$ is the difference between the weight of a minimum spanning tree of G and the weight of a constrained minimum spanning tree of G in which the presence of edge e is enforced.*

The relative costs can be efficiently computed once for all. We start by the application of Kruskal's algorithm (Kruskal, 1956) to build a minimum spanning tree T_{MST} of G in time $O(|E||V|)$. The relative cost of every edge $e \in T_{MST}$ is $r_e = 0$. Otherwise, $r_e = c_e - c_{e'}$, where e' is the edge with maximum weight among those in the cycle generated by inserting e into T_{MST}. The rationale for the use of the relative cost comes from a property defined in (Chandy and Lo, 1973; Chandy and Russel, 1972):

PROPERTY 1. *Let T be an optimal solution to CMST and T_{MST} a minimum spanning tree of the same graph. If the nodes i and j are in the same component of T and edge $e = (i, j)$ belongs to T_{MST}, then this edge also belongs to T.*

The SR (saturation and relative cost) test checks if the components of each extremity of each candidate edge $e = (i, j) \in E$ are saturated and do not have any strictly positive reduced cost edge, except for those connecting them to the central node r. Candidate edges satisfying this condition are discarded, since they are unlikely to be part of an improving move. This test can be implemented in $O(1)$ time for every candidate edge, using the information already obtained by depth-first search.

The other reduction tests are based on the decomposition of the cost of a move defined by a candidate edge $e = (u, v) \in E$ for insertion and by a candidate path $(p-q)$ for removal (whose extremal vertices are p and q). Without loss of generality, we assume that p (resp. q) belongs to the same component as u (resp. v), as shown in Figure 3. The cost of applying this move to the current solution T to obtain a neighbor T' can be computed as $\Delta = \sum_{e \in T'} c_e - \sum_{e \in T} c_e$. This value can be rewritten as $\Delta = \Delta_1 + \Delta_2$, where Δ_1 is associated with the edge exchanges that will generate the component containing edge e in the new solution T' and Δ_2 addresses the reconnection of the remaining elements along the candidate path $(p-q)$.

The computation of Δ_1 is based on four particular edges, as illustrated in Figure 3. We first define $\hat{T}$ as the tree containing edge e in the forest resulting from the removal of path $(p-q)$. Let e_1 (resp. e_2) be the first edge in the component shared by the paths going from u and p (resp. v and q) to the

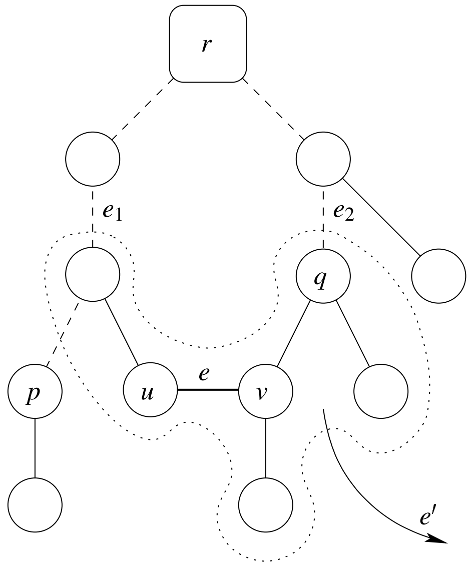


Figure 3. Selection of a candidate edge

central node r. Among those in the path $(p-q)$, edges e_1 and e_2 are the most appropriate be involved in the computation of Δ_1. The removal of edges between e_1 and e_2 (e.g. edges incident to r in Figure 3) would disconnect nodes of T which do not belong to $\hat{T}$. The removal of other edges (e.g. edges incident to p in Figure 3) do not disconnect $\hat{T}$ from T. We now consider the forest obtained once path $(p-q)$ has been removed. After reconnecting the forest and applying the savings heuristic EW, at least one edge e' will be incident to $\hat{T}$ in the new solution T'. Then, $\Delta_1 = c_e + c_{e'} - c_{e_1} - c_{e_2}$. The other component Δ_2 addresses the cost of first disconnecting, then reconnecting all other subtrees affected by the deletion of path $(p-q)$.

We now introduce the reduction tests LB1 and LB2. Both of them discard the cost Δ_2 and make use of a lower bound $\underline{\Delta}_1 \leqslant \Delta_1$. Indeed, a candidate edge e can be eliminated if $\underline{\Delta}_1 \geqslant 0$ for every possible candidate path $(p-q)$ that can be used in conjunction with e. Test LB1 is based on a weak estimation of the weights of the three actual edges e', s_1, and s_2 to compute $\underline{\Delta}_1$ in time $O(1)$. The reconnection cost $c_{e'}$ can be underestimated by the minimum edge weight $\underline{c}$ over all edges of the graph. The disconnection cost c_{s_1} (resp. c_{s_2}) can be overestimated by the largest edge weight $\underline{c}_u$ (resp. $\underline{c}_v$) among those in the path from vertex u (resp. v) to the central vertex. The bound $\underline{\Delta}_1 = c_e + \underline{c} - \underline{c}_u - \underline{c}_v$ can be computed in time $O(1)$, since the computation of the value $\underline{c}_u$ for every $u \in V$ can be performed along the application of depth-first search.

Test LB2 is based on a stronger estimation of the weights of the three actual edges e', e_1, and e_2. The disconnection cost c_{e_1} (resp. c_{e_2}) can be exactly computed in time $O(Q)$. The reconnection cost $c_{e'}$ can be underestimated by the minimum edge weight among all those incident to the new component containing e. This underestimation can be computed in time $O(Q)$ if the incident edges to each node are ordered by increasing weights. Test LB1 is clearly dominated by LB2. However, the computation of LB1 is still of interest to perform a first round of quick reductions. Test LB2 is applied afterwards, to the candidates that passed test LB1.

2.2. Selection of path extremities

Given a candidate edge e considered for insertion, there are $O(Q^2)$ possible choices for the associated path $(p-q)$ involved in the move. Each move can be evaluated in time $O(|V|^2 \log |V|)$ using the EW procedure. Substantial savings in computation time can be achieved by evaluating only a small fraction of these paths. We propose heuristic rules to restrict the exact move evaluation to only two paths for each candidate edge e.

We first consider the selection of the first path, as illustrated in Figure 4. Given a candidate edge $e = (u,v) \in E$, we indentify in time $O(Q)$ the nodes belonging to the paths from u and v to the central node. We start by applying a depth-first search to the component containing u. For each node visited along the depth-first search, another depth-first search is applied to the component containing v. Let p (resp. q) be the node being visited by depth-first search in the component containing u (resp. v). The path connecting p and q may be removed if the tree containing $e = (u,v)$ in the resulting forest does not violate the capacity constraint. If p is in the path from u to the central node, this move will carry to the component containing e in the new solution all nodes currently in the subtree rooted at p. Otherwise, let p' be the node in the path from p to the central node whose predecessor p'' belongs to the path from u to the central node. In this case, the move will carry all nodes in the subtree rooted at p'', except those also belonging to the subtree rooted at p'. A $(p-q)$ path is said to be feasible if the total weight of the nodes carried from the components containing u and v is less than or equal to Q. Since the capacities of each subtree are systematically computed along the application of depth-first search at the beginning of each local search iteration, the feasibility of each path can be evaluated in time $O(1)$.

Once a feasible $(p-q)$ path has been identified, we can compute an estimation of the reconnection cost. Let c_p (resp. c_q) be the weight of the minimum weight edge incident to p (resp. q) among all those that can be used to reconnect the subtree rooted at p (resp. q) either to the central node or to another component not involved in the move, without violating the capacity constraints. If nodes p and q are both in the tree containing e in the forest

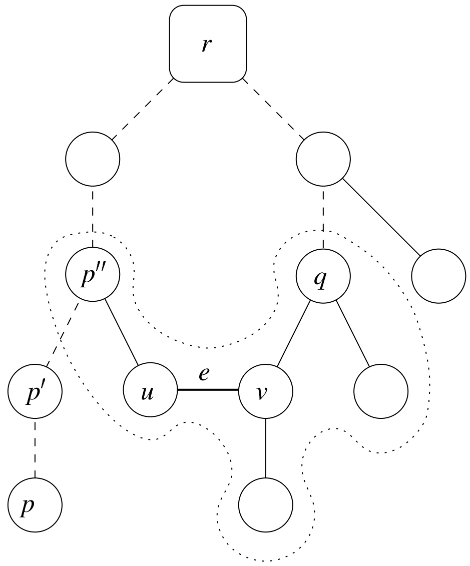


Figure 4. Selection of path extremities

obtained by the removal of path $(p-q)$, the estimation of the reconnection costs is given by $\min\{c_p, c_q\} + c_e$. Otherwise, it is given by $c_p + c_q$. These computations can be performed in time $O(|V|)$, since all edges incident to p (resp. q) may have to be inspected.

The first selected path is that with the smallest estimation of the reconnection costs (ties are randomly broken). To avoid the repeated selection of the same path when different candidate edges connect the same components, a second path for each candidate edge is always randomly chosen.

2.3. Local search procedure

In this section we summarize the main steps of the local search procedure for CMST. Figure 5 gives the algorithmic description of this procedure, which receives an initial solution T_0 as a parameter. We denote by $c(T)$ the cost of any feasible solution T. The current solution T is initialized with the starting solution T_0. The external loop is performed as long as improvements in the current solution are possible. The best neighbor $\bar{T}$ of T is initialized with T itself. Every edge $e \in E$ with extremities in two different components of T is considered as a candidate for insertion into the current tree. An edge e is discarded if it fails to pass any of the tests SR, LB1, and LB2. Tests SR and LB1 are applied first, since their complexity is smaller.

In case a candidate edge e is not eliminated by the reduction tests, two possible paths $(p_1 - q_1)$ and $(p_2 - q_2)$ are considered for removal, as described in Section 2.2. $\bar{T}$ is eventually substituted by the best among the two solutions obtained by inserting e and removing either path $(p_1 - q_1)$ or path $(p_2 - q_2)$. Finally, the current solution T is replaced by $\bar{T}$ in case an improvement was detected, and a new local search iteration resumes.

```
procedure LS_CMST(T0);
  T ← T0;
  improvement ← .TRUE.;
  while improvement do;
      improvement ← .FALSE.;
      T̄ ← T;
      for each edge e ∈ E with extremities in different components do;
          if edge e satisfies tests SR, LB1, and LB2 then do;
              Select the candidate paths (p1 − q1) and (p2 − q2);
              Obtain T′ by performing the move defined by e and (p1 − q1);
              if c(T′) < c(T̄) then T̄ ← T′;
              Obtain T′ by performing the move defined by e and (p2 − q2);
              if c(T′) < c(T̄) then T̄ ← T′;
              if c(T̄) < c(T) then do;
                  T ← T̄;
                  improvement ← .TRUE.;
              end if;
          end if;
      end for;
  end while;
  return T;
end LS_CMST;
```

Figure 5. Local search procedure

3. Effectiveness of the local search strategy

The local search algorithm described in the previous section was implemented in C, using version 2.8.1 of the gcc compiler for AIX, with the optimization flag set to -O. All computational experiments were performed on a 200 MHz RISC 6000 workstation model 43P-140 with 64 Mbytes of RAM memory.

We report numerical results obtained by our heuristic on a subset of the test problems described by Gouveia and Martins (2000). The instances have 40 and 80 nodes, besides the central node. They are divided into two classes, in which the central node is placed either in the middle (tc) or at one of the corners (te) of an 100×100 grid. The nodes are randomly generated within this grid and edge costs are equal to the integer part of the Euclidean distance between their extremities. All node weights are equal to one. For each class there are ten graphs with 40 nodes and five with 80 nodes. For each graph with 40 nodes, there are two different instances with $Q = 5$ and $Q = 10$. For each of the larger graphs with 80 nodes, there are three instances with $Q = 5$, $Q = 10$, and $Q = 20$. Optimal solutions are known for all problems with 40 nodes and for 14 out of the 30 problems with 80 nodes. For the 16 remaining problems for which an optimal solution is not known, the best known upper bounds are within less than 2% of the lower bound described in (Gouveia and Martins, 2000).

Table I. Computational results: local search on problems with 40 nodes (central node in the middle)

		Esau-Williams			Complete LS			Reduced LS		
Instance	Q	best %	avg. %	s	best %	avg. %	s	best %	avg. %	s
tc40-1	5	0.68	1.89	0.0	0.00	0.77	2.1	0.00	1.59	0.0
tc40-2	5	1.73	3.39	0.0	0.00	0.67	2.2	0.00	1.57	0.0
tc40-3	5	2.25	2.63	0.0	0.00	0.26	2.1	0.00	0.45	0.0
tc40-4	5	1.62	2.92	0.0	0.00	0.10	2.5	0.00	0.42	0.1
tc40-5	5	0.83	3.00	0.0	0.50	0.93	1.6	0.83	1.97	0.0
tc40-6	5	1.53	2.78	0.0	0.00	0.00	2.3	0.85	1.39	0.0
tc40-7	5	0.00	2.78	0.0	0.00	0.10	2.1	0.00	0.10	0.1
tc40-8	5	0.90	1.79	0.0	0.36	0.36	1.9	0.36	0.94	0.0
tc40-9	5	5.18	6.78	0.0	0.00	1.79	1.8	0.00	2.42	0.1
tc40-10	5	0.00	3.15	0.0	0.00	0.17	2.9	0.00	1.52	0.0
tc40-1	10	0.00	1.53	0.0	0.00	0.00	22.0	0.00	0.16	0.0
tc40-2	10	0.00	1.18	0.0	0.00	0.00	12.2	0.00	0.49	0.0
tc40-3	10	1.60	3.40	0.0	1.60	2.00	18.4	1.60	2.08	0.0
tc40-4	10	0.00	1.45	0.0	0.00	0.00	27.4	0.00	0.00	0.1
tc40-5	10	0.00	0.00	0.0	0.00	0.00	13.4	0.00	0.00	0.0
tc40-6	10	0.80	0.80	0.0	0.00	0.00	26.0	0.00	0.20	0.0
tc40-7	10	0.00	1.04	0.0	0.00	0.00	19.2	0.00	0.00	0.0
tc40-8	10	0.00	0.00	0.0	0.00	0.00	10.7	0.00	0.00	0.0
tc40-9	10	0.00	0.99	0.0	0.00	0.00	16.6	0.00	0.08	0.0
tc40-10	10	1.93	3.13	0.0	0.00	0.12	32.2	0.00	1.41	0.1

Tables I to III show the computational results obtained over ten runs of each instance. For each instance, we first give the minimum and the average deviations in percent of the solution obtained by the savings heuristic EW, with respect to the best known values given in (Gouveia and Martins, 2000), together with the average computation time in seconds. The same information

Table II. Computational results: local search on problems with 40 nodes (central node at one of the corners)

		Esau-Williams			Complete LS			Reduced LS		
Instance	Q	best %	avg. %	s	best %	avg. %	s	best %	avg. %	s
te40-1	5	2.77	4.59	0.0	0.24	1.18	2.3	1.33	2.90	0.1
te40-2	5	1.77	3.96	0.0	0.00	1.64	2.1	0.51	2.90	0.1
te40-3	5	2.89	4.28	0.0	0.00	1.23	2.6	0.00	1.63	0.2
te40-4	5	3.07	4.78	0.0	0.00	1.44	2.4	0.12	2.67	0.2
te40-5	5	1.40	2.91	0.0	0.26	0.31	2.1	0.51	0.82	0.1
te40-6	5	0.61	4.34	0.0	0.00	1.67	2.6	0.00	1.80	0.2
te40-7	5	2.07	4.63	0.0	0.73	1.28	1.9	0.85	1.66	0.2
te40-8	5	4.72	5.11	0.0	0.12	0.27	2.5	1.09	1.78	0.1
te40-9	5	3.08	3.18	0.0	0.13	0.13	2.1	0.13	0.27	0.1
te40-10	5	1.68	2.42	0.0	0.52	0.52	2.3	0.52	0.76	0.2
te40-1	10	2.85	7.57	0.0	0.34	2.10	41.2	1.01	4.75	0.3
te40-2	10	5.58	7.96	0.0	0.00	1.26	49.9	0.00	2.67	0.6
te40-3	10	2.11	5.16	0.0	0.00	0.97	39.2	0.35	1.90	0.4
te40-4	10	0.67	0.67	0.0	0.00	0.00	19.5	0.34	0.34	0.2
te40-5	10	3.67	3.67	0.0	0.00	0.00	50.0	0.00	0.93	0.3
te40-6	10	2.43	7.29	0.0	0.00	0.62	38.4	0.00	5.12	0.3
te40-7	10	3.38	7.53	0.0	2.88	2.88	20.1	2.88	3.67	0.1
te40-8	10	4.92	5.25	0.0	0.66	0.66	43.9	0.98	1.90	0.2
te40-9	10	4.27	8.75	0.0	0.18	0.18	35.8	0.18	2.85	0.4
te40-10	10	3.83	5.66	0.0	0.00	1.88	44.4	1.28	2.76	0.5

is given for solutions obtained by the local search procedure described in the previous section. To illustrate the effectiveness of the neighborhood reduction strategy, we give results for two variants: without neighborhood reduction (complete LS) and with neighborhood reduction (reduced LS).

The results in these three tables show that the local search with neighborhood reduction significantly improves the solutions found by the savings procedure. The average improvements are summarized in Table IV. For each class and for each value of the instance size (40 nodes and 80 nodes), we give the average percentual reduction in the deviation from the best known solution obtained by the complete and the reduced local search procedures, with respect to the savings heuristic. Over the 70 instances, the latter found nine best known solutions, while the complete and reduced local search procedures obtained respectively 38 and 23. In particular, the savings procedure did not find any best known solution for the harder problems with the central

Table III. Computational results: local search on problems with 80 nodes

		Esau-Williams			Complete LS			Reduced LS		
Instance	Q	best %	avg. %	s	best %	avg. %	s	best %	avg. %	s
tc80-1	5	5.91	7.36	0.0	0.45	1.90	32.2	2.46	4.15	0.3
tc80-2	5	5.36	6.36	0.0	1.64	1.96	31.9	2.00	4.27	0.2
tc80-3	5	4.29	6.00	0.0	0.00	0.84	35.1	1.68	3.13	0.3
tc80-4	5	5.46	6.66	0.0	0.46	1.64	38.2	3.15	4.55	0.2
tc80-5	5	3.03	6.04	0.0	0.08	0.95	31.4	2.10	3.43	0.4
tc80-1	10	4.84	6.27	0.0	0.00	0.82	593.2	0.90	2.38	0.7
tc80-2	10	2.05	4.50	0.0	0.91	1.46	665.2	1.60	2.20	0.7
tc80-3	10	2.73	5.26	0.0	0.00	0.64	466.0	0.46	2.74	0.5
tc80-4	10	5.41	6.19	0.0	0.92	1.61	676.5	3.23	4.49	0.3
tc80-5	10	5.39	6.97	0.0	0.20	1.77	477.1	0.30	2.66	0.8
tc80-1	20	0.96	2.58	0.0	0.00	0.24	4739.2	0.48	0.70	0.4
tc80-2	20	1.95	2.84	0.0	0.00	0.80	3289.0	0.73	1.27	0.6
tc80-3	20	0.48	1.30	0.0	0.00	0.19	2811.2	0.48	0.92	0.2
tc80-4	20	1.22	2.34	0.0	0.00	0.61	3219.4	0.00	1.34	0.4
tc80-5	20	3.49	3.49	0.0	0.00	0.61	5088.4	0.87	2.54	0.8
te80-1	5	2.91	3.20	0.0	1.06	1.30	21.0	0.94	1.70	1.5
te80-2	5	1.49	2.69	0.0	0.20	1.09	14.1	0.55	1.52	1.0
te80-3	5	2.60	3.12	0.0	0.57	1.18	21.4	0.88	1.63	1.8
te80-4	5	2.35	2.97	0.0	0.59	1.04	15.3	1.13	1.54	1.2
te80-5	5	2.07	3.09	0.0	0.00	0.88	17.2	0.73	2.07	0.5
te80-1	10	3.56	3.74	0.0	0.06	1.36	321.4	1.39	2.01	4.6
te80-2	10	2.81	4.08	0.0	1.65	2.25	189.5	2.56	3.56	0.9
te80-3	10	3.68	5.05	0.0	0.77	1.13	511.5	0.95	2.03	5.4
te80-4	10	7.55	9.15	0.0	1.72	2.56	662.4	2.39	6.17	4.7
te80-5	10	6.55	6.87	0.0	0.44	0.51	570.3	4.80	5.44	1.2
te80-1	20	0.78	3.14	0.0	0.16	0.75	4678.4	0.78	1.30	6.7
te80-2	20	5.23	7.71	0.0	0.16	1.58	5132.1	0.74	4.12	10.7
te80-3	20	5.13	6.57	0.0	0.00	0.40	5745.7	0.08	0.85	11.5
te80-4	20	7.59	8.23	0.0	0.16	0.61	4614.8	0.40	1.52	15.8
te80-5	20	0.89	2.15	0.0	0.00	0.02	3641.0	0.00	0.40	4.8

node at one of the corners of the grid, while the complete and reduced local search procedures found respectively 13 and six best known solutions for the 30 problems in this class.

Table IV. Average reductions with respect to the savings heuristic

Instances	Complete LS (%)	Reduced LS (%)
tc40	1.87	1.39
te40	3.99	2.80
tc80	3.93	2.28
te80	3.89	2.61

The computation times observed with the neighborhood reduction strategies are significantly smaller, without too much deterioration in solution quality. For the problems with $n = 80$, $Q = 10$, and the central node at one of the corners of the grid, Figure 6 illustrates the effect of tests SR, LB1, and LB2 described in Section 2.1. For each instance, we give the fraction in percent of the number of candidate edges (i.e., those edges with both extremities in different components) which are not eliminated by each combination of tests (SR alone, LB1 alone, SR together with LB1, and LB2 after SR and LB1). None of the $O(1)$ tests SR and LB1 dominate the other. They eliminate at least 37% of the candidate edges of any of the five instances in this figure. The fraction of remaining edges range from 10 to 30% after the application of the three tests.

For the same problems, Figure 7 shows the fraction of the iterations in which the best movement was that obtained with the path choosed by the first criterion, by the random choice, or by both of them. It illustrates that none of the criteria is dominated by the other and that both should be used.

The results in this section show that the reduced local search procedure compares favourably with the savings heuristic of Esau and Williams. Local search obtains much better solutions in marginal time, due to the effectiveness of the reduction procedures. Therefore, the reduced local search procedure represents a good trade-off between solution quality and computation time. In the next section, we describe an extension of this approach, embedding the reduced local search strategy into a GRASP heuristic.

4. GRASP with a memory-based local search strategy

A GRASP is a multistart process, in which each iteration consists of two phases: construction and local search (Feo and Resende, 1995). The best

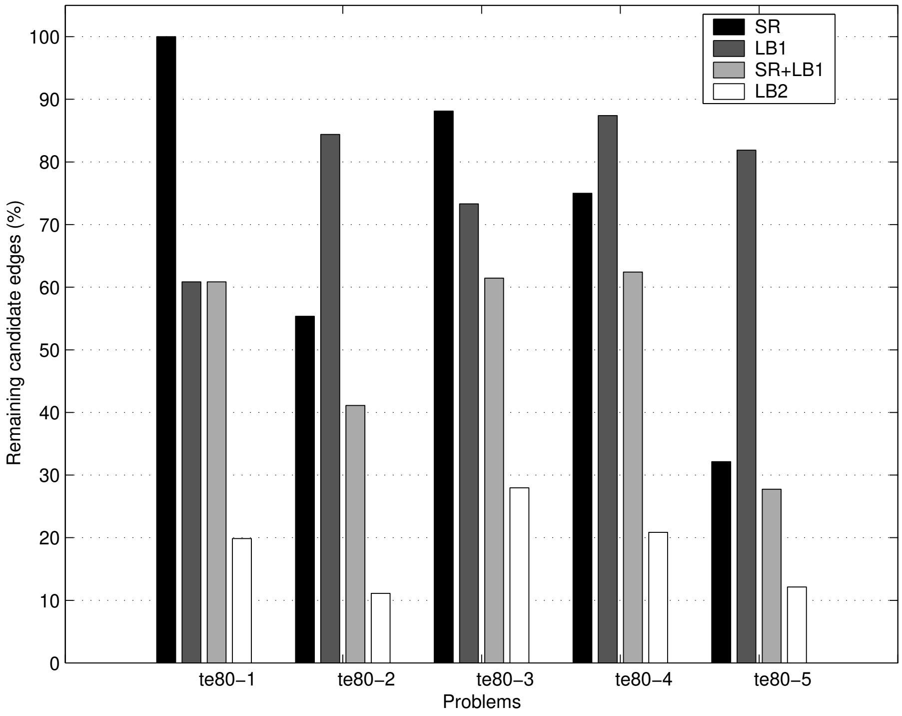


Figure 6. Effectiveness of the reductions tests SR, LB1, and LB2

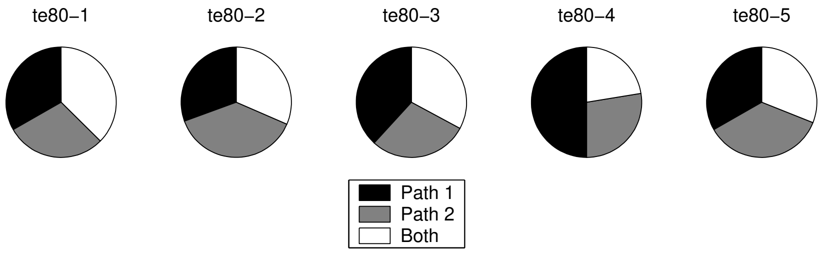


Figure 7. Path selection criteria

solution found after `Max_Iterations` iterations is returned. The reader is refered to Festa and Resende (2002) for a recent survey of applications and to Resende and Ribeiro (2001b) for implementations strategies, variants, and extensions.

A feasible solution is iteratively constructed in the first phase, one element at a time. At each construction iteration, the choice of the next element to be added is determined by ordering all candidate elements (i.e. those that can be added to the solution) in a candidate list, with respect to its contribution to the objective function. The list of best candidates is called the *restricted candidate list* (RCL). The random selection of an element from the RCL allows for different solutions to be obtained at each GRASP iteration. Since solutions generated at the construction phase are not guaranteed to be locally optimal, it is almost always beneficial to apply a local search as an attempt to improve each constructed solution.

In the remainder of this section, we customize a GRASP for the capacitated minimum spanning tree problem. We describe the construction and local search procedures, as well as a path-relinking intensification strategy.

4.1. CONSTRUCTION PHASE

The construction phase is a randomized version of the savings heuristic of Esau and Williams. It starts with an r-centered star, in which all nodes are directly connected to the central node. Each iteration is characterized by the merging of two components into a new one. Two components may be merged if the sum of the weights of their nodes is less than or equal to Q. This operation is performed in two steps. First, we select an edge $(i,j) \in E$ connecting nodes i and j from two different components T_i and T_j. Let i' (resp. j') be the node connecting T_i (resp. T_j) to the central node r. Next, we eliminate the maximum weight edge among the two that connect T_i and T_j to the central node. This operation leads to saving $s_{ij} = \max\{c_{i'r}, c_{j'r}\} - c_{ij}$ cost units with respect to the current solution. Let C be the subset formed by all edges $(i,j) \in E$ whose associated components T_i and T_j can be merged. For a given parameter value $\alpha \in [0,1]$ (we used $\alpha = 0.9$ in our implementation), we insert in the RCL all edges from C with savings $s_{ij} \geq s^{min} + \alpha \cdot (s^{max} - s^{min})$, where $s^{min} = \min\{s_{ij} > 0 : (i,j) \in C\}$ and $s^{max} = \max\{s_{ij} > 0 : (i,j) \in C\}$. An edge is randomly selected from the RCL, the components associated to its extremities are merged, and a new iteration resumes, until no further savings are possible.

4.2. LOCAL SEARCH PHASE

The local search phase seeks to improve each solution built in the construction phase. Basically, the same strategy described in Section 2 is used.

As a memoryless approach, the basic GRASP procedure does not make use of the information collected along the search. Also, the local search phase of a GRASP iteration very often repeatedly spends too much time to stop in the first same local optimum, eventually missing better, but hidden solutions which might be quite close to this local optimum. As an attempt to improve the effectiveness of the local search phase, we propose to add to the latter some elements of very short term memory taken from tabu search (Glover and Laguna, 1997).

However, instead of maintaining a list of forbidden attributes, we keep the last `Tabu_Tenure` visited solutions in the memory. We always move to the best neighbor which is not stored in this memory, regardless of whether it is an improving solution or not. The search stops if the incumbent is not improved after a sequence of `Tabu_Tenure` local search iterations. A small value is used for this parameter, to keep the computation times limited.

4.3. PATH-RELINKING

Path-relinking was originally proposed by Glover (1996), as an intensification strategy exploring trajectories connecting elite solutions obtained by tabu search or scatter search (Glover and Laguna, 1997; Glover, 2000; Glover et al., 2000). The use of path-relinking within a GRASP procedure was first proposed by Laguna and Martí (1999), followed by several extensions, improvements, and successful applications (Aiex et al., 2000; Canuto et al., 2001; Resende and Ribeiro, 2001a; Ribeiro et al., 2001). Implementation strategies are investigated in detail by Resende and Ribeiro (2001b).

In this context, path-relinking is applied to pairs of solutions, one obtained by local search and the other randomly chosen from a pool with a limited number `Max_Elite` of elite solutions found along the search. The pool is originally empty. Each locally optimal solution is considered as a candidate to be inserted into the pool if it is different from every other solution currently in the pool. If the pool already has `Max_Elite` solutions and the candidate is better than the worst of them, then the former replaces the latter. If the pool is not full, the candidate is simply inserted.

Let z (resp. y) be the an elite solution in the pool (resp. a locally optimal solution). Resende and Ribeiro (2001b) observed that better solutions are usually found in smaller computation times if z is used as the initial solution, and y as the guiding one. The algorithm starts by computing the edges that belong to either y or z, but not to the other. These computations amount to determining a set of moves $\Delta(z,y)$ which should be applied to the initial solution to reach the guiding one. Each move is characterized by a pair of edges, one to be inserted and the other to be eliminated from the current solution. The best solution $\bar{y}$ along the new path to be constructed is initialized with the initial solution. Each path-relinking iteration has three steps: the best remaining move leading to a feasible tree is applied to the current solution, the incumbent $\bar{y}$ is updated, and the selected move is removed from $\Delta(z,y)$. The algorithm stops when the guiding solution is reached or in case all remaining moves lead to infeasible solutions. The incumbent $\bar{y}$ is returned as the best solution found by path-relinking and inserted into the pool if it satisfies the membership conditions.

We recall that we denote by $c(x)$ the weight of a solution x. The pseudocode with the complete description of the procedure `GRASP+PR_CMST` for the capacitated minimum spanning tree problem is given in Figure 8. This description incorporates the construction, local search, and path-relinking phases.

```
procedure GRASP+PR_CMST;
  c* ← ∞;
  Pool ← ∅;
  for k = 1,...,Max_Iterations do;
      Construct a greedy randomized solution x;
      Find y by applying local search to x;
      if y satisfies the membership conditions then insert y into Pool;
      Randomly select a solution z ∈ Pool (z ≠ y) with uniform probability;
      if c(z) > c(y) then y ↔ z;
      Compute Δ(z,y);
      Let ȳ be the best solution found by applying path-relinking to (y,z);
      if ȳ satisfies the membership conditions then insert ȳ into Pool;
      if c(ȳ) < c* then do;
          x* ← ȳ;
          c* ← c(ȳ);
      end if;
  end for;
  return x*;
end GRASP+PR_CMST;
```

Figure 8. Pseudo-code of the GRASP with path-relinking procedure for the capacitated minimum spanning tree problem

5. Computational results

We report in this section the numerical results obtained with the algorithm `GRASP+PR_CMST` described in the previous section. This heuristic was also implemented in `C`, using version 2.8.1 of the `gcc` compiler for AIX, with the optimization flag set to `-O`. Again, all experiments were performed on a 200 MHz RISC 6000 workstation model 43P-140 with 64 Mbytes of RAM memory.

In addition to the test problems already described in Section 3, we considered some larger instances with 120 and 160 nodes, besides the central node, also described by Gouveia and Martins (2000). These instances are generated as the previous ones. They are also divided into two classes, in which the central node is placed either in the middle (tc) or at one of the corners (te) of an 100×100 grid. There is only one graph with 120 nodes and another one with 160 nodes for each class: tc-120, te-120, tc-160, and te-160. There are three different instances in each case, with $Q = 5$, $Q = 10$, and $Q = 20$.

The computational experiments reported in (Amberg et al., 1996) and (Ahuja et al., 2001) used instances with 40 and 80 nodes. Amberg et al. (1996) considered 60 instances: those indexed from 1 to 5 with 40 nodes and all instances with 80 nodes. The results presented by Ahuja et al. (2001) comprise 65 instances: all those with 40 nodes and the central node in the middle, those indexed from 1 to 5 with 40 nodes and the central node at one of the corners, and all instances with 80 nodes. Results for the instances with 120 nodes are reported by Gouveia and Martins (2000), while those for the instances with 160 nodes appeared exclusively in (Martins, 1999).

The best known solutions values (i.e., the best known upper bounds) for all problems with 40, 80, and 120 nodes are reported in (Gouveia and Martins, 2000). The best known solution values for instance te120 with $Q = 20$ and for all instances with 160 nodes were informed by (Martins, 2001). Amberg et al. (1996) found the best known upper bounds for 67 out of the 70 instances with 40 and 80 nodes, using three variants of tabu search (not all best upper bounds are found by the same variant). Ahuja et al. (2001) improved the previous results for three instances with 80 nodes, finding the best known upper bounds for all these instances using the same tabu search algorithm.

We report computational results for all test instances with 40 to 160 nodes. One possible shortcoming of the standard memoryless GRASP metaheuristic is the independence of its iterations, i.e., the fact that it does not learn from the history of solutions found in previous iterations. This is so because it discards information about any solution that does not improve the incumbent. However, the use of a very short term memory in an extended local search procedure can improve the solutions found, by adding some elements of tabu search, as proposed in Section 4.2. Also, long term memory information gathered from good solutions can be used to implement extensions based on path-relinking, as described in Section 4.3. With these extensions, we were able to build a GRASP heuristic using a memory-based local search incorporating short- and long-term memory features. To illustrate the effectiveness of these two extensions of the basic algorithm, we report in Tables V to VIII the results obtained for both the memoryless algorithm and the full procedure `GRASP+PR_CMST` with path-relinking and short term memory. For each instance and for each approach, we give the deviation in percent of the best solution value found with respect to the best known upper bound, the GRASP iteration in which this solution was found, and the total computation time in seconds. Parameters `Max_Iterations`, `Max_Elite`, and `Tabu_Tenure` were set once for all and without tuning, respectively at 200, 20, and 6.

Procedure `GRASP+PR_CMST` improved many solutions found by the memoryless approach, finding the optimal values for all instances with 40 nodes.

Table V. Computational results: memoryless GRASP vs. GRASP with path-relinking and short term memory on problems with 40 nodes (central node in the middle)

		Memoryless GRASP			GRASP+PR_CMST		
Instance	Q	deviation %	iteration	s	deviation %	iteration	s
tc40-1	5	0.00	181	9.9	0.00	1	51.4
tc40-2	5	0.00	6	13.5	0.00	4	42.2
tc40-3	5	0.00	7	8.1	0.00	1	30.4
tc40-4	5	0.00	13	16.4	0.00	2	48.8
tc40-5	5	0.33	90	15.0	0.00	15	50.0
tc40-6	5	0.00	1	17.1	0.00	1	64.9
tc40-7	5	0.00	2	23.7	0.00	2	103.1
tc40-8	5	0.00	43	9.0	0.00	1	45.4
tc40-9	5	0.00	1	14.0	0.00	1	53.3
tc40-10	5	0.00	3	21.1	0.00	1	72.6
tc40-1	10	0.00	13	8.4	0.00	4	46.8
tc40-2	10	0.00	1	10.1	0.00	1	25.8
tc40-3	10	0.00	53	11.1	0.00	3	53.4
tc40-4	10	0.00	4	27.8	0.00	2	110.0
tc40-5	10	0.00	30	14.2	0.00	1	60.2
tc40-6	10	0.00	14	17.5	0.00	3	68.5
tc40-7	10	0.00	19	17.3	0.00	4	65.3
tc40-8	10	0.00	3	14.1	0.00	1	69.7
tc40-9	10	0.00	8	6.9	0.00	1	41.0
tc40-10	10	0.00	6	22.8	0.00	2	112.8

Although the overall computation times increased, we notice that the best solutions were found much earlier. The best solution is found at the first iteration for 13 instances and in at most 20 iterations for 37 instances, among the 40 problems in Tables V and VI. We recall that the computation times are reported for a total of 200 iterations. Similar results are observed in Table VII for the test problems with 80 nodes. The improved GRASP procedure missed only four of the best known upper bounds (we recall that Amberg et al. (1996) missed three upper bounds, but using three different algorithms). In these few cases, the deviation from the best upper bound was always less than 0.60%. For more than 50% of the problems with results reported in this table, the best solution is found in at most 20 iterations.

The instances with 120 and 160 nodes were run on a Celeron 800 MHz machine with 128 Mbytes of RAM memory. The program was compiled with version 2.96 of the `gcc` compiler for Linux, once again with the optimization flag set to `-O`. The computational results for these larger instances are given in

Table VI. Computational results: memoryless GRASP vs. GRASP with path-relinking and short term memory on problems with 40 nodes (central node at one of the corners)

		Memoryless GRASP			GRASP+PR_CMST		
Instance	Q	deviation %	iteration	s	deviation %	iteration	s
te40-1	5	0.00	35	78.3	0.00	13	208.2
te40-2	5	0.00	8	53.8	0.00	6	176.7
te40-3	5	0.00	61	69.4	0.00	2	215.0
te40-4	5	0.00	84	58.7	0.00	5	211.7
te40-5	5	0.00	6	58.7	0.00	5	189.2
te40-6	5	0.00	30	66.2	0.00	9	178.6
te40-7	5	0.24	20	67.1	0.00	19	215.4
te40-8	5	0.00	25	62.2	0.00	8	179.8
te40-9	5	0.13	3	67.4	0.00	24	195.3
te40-10	5	0.13	125	60.5	0.00	50	192.5
te40-1	10	0.34	18	177.7	0.00	13	549.8
te40-2	10	0.00	1	132.7	0.00	1	489.3
te40-3	10	0.35	40	135.8	0.00	28	410.0
te40-4	10	0.00	18	105.2	0.00	2	382.4
te40-5	10	0.00	13	104.0	0.00	1	272.1
te40-6	10	0.00	32	125.9	0.00	2	340.1
te40-7	10	0.34	3	108.2	0.00	1	423.0
te40-8	10	0.33	12	160.6	0.00	17	446.5
te40-9	10	0.00	23	107.8	0.00	2	334.1
te40-10	10	0.00	177	124.7	0.00	3	389.5

Table VIII. Besides the statistics given in the previous tables, we also inform the currently best known solution value according with (Martins, 2001) and the new best solution value for the instances in which the latter was improved by the GRASP heuristic with the memory-based local search. Once again, we notice that algorithm `GRASP+PR_CMST` improved the solutions found by the basic GRASP procedure for all 12 instances in this table. In particular, the GRASP heuristic with the memory-based local search improved the best known solution for five out of these 12 instances (tc160 with $Q = 5$, $Q = 10$, and $Q = 20$; te160 with $Q = 10$ and $Q = 10$). For the other problems, the deviation from the best upper bound was always less than 0.25%.

Table VII. Computational results: memoryless GRASP vs. GRASP with path-relinking and short term memory on problems with 80 nodes

		Memoryless GRASP			GRASP+PR_CMST		
Instance	Q	deviation %	iteration	s	deviation %	iteration	s
tc80-1	5	0.36	105	154.6	0.00	7	325.4
tc80-2	5	0.73	33	165.8	0.09	159	401.5
tc80-3	5	0.00	161	126.0	0.00	8	294.2
tc80-4	5	0.93	191	134.9	0.56	56	309.0
tc80-5	5	0.08	107	195.1	0.00	122	432.8
tc80-1	10	0.45	14	214.7	0.00	14	448.3
tc80-2	10	1.37	49	346.4	0.00	28	785.0
tc80-3	10	0.69	8	188.0	0.23	3	390.1
tc80-4	10	0.92	5	222.9	0.00	49	517.5
tc80-5	10	1.00	45	368.3	0.00	52	820.7
tc80-1	20	0.00	47	136.1	0.00	2	239.7
tc80-2	20	0.73	57	227.5	0.00	65	571.5
tc80-3	20	0.00	5	87.3	0.00	2	187.4
tc80-4	20	0.49	46	151.2	0.00	2	262.3
tc80-5	20	0.22	169	421.8	0.00	2	1023.5
te80-1	5	0.08	44	802.5	0.00	83	1609.2
te80-2	5	0.39	98	940.6	0.00	81	1938.4
te80-3	5	0.31	41	1039.9	0.00	88	2228.2
te80-4	5	0.51	192	987.3	0.08	55	2128.9
te80-5	5	0.00	111	741.2	0.00	19	1658.0
te80-1	10	0.06	183	2480.9	0.00	5	5387.8
te80-2	10	0.37	36	2528.5	0.00	21	6172.2
te80-3	10	0.36	117	2825.2	0.00	4	6088.5
te80-4	10	0.43	12	2677.6	0.00	5	6311.5
te80-5	10	0.00	106	2434.5	0.00	39	5465.1
te80-1	20	0.15	191	3582.3	0.00	15	6423.5
te80-2	20	0.00	65	5040.1	0.00	11	11034.1
te80-3	20	0.24	80	5139.1	0.00	79	10608.2
te80-4	20	0.00	79	4632.7	0.00	2	10038.2
te80-5	20	0.00	15	3039.2	0.00	2	5676.7

Table VIII. Computational results: memoryless GRASP vs. GRASP with path-relinking and short term memory on problems with 120 and 160 nodes

		Memoryless GRASP			GRASP+PR_CMST			Best value	
Instance	Q	dev. %	iter.	s	dev. %	iter.	s	previous	new
tc120	5	0.23	185	342.3	0.00	64	665.4	1291	same
tc120	10	0.22	55	1111.1	0.00	3	2368.0	904	same
tc120	20	0.52	23	1314.4	0.00	4	3075.5	768	same
te120	5	0.59	7	1323.3	0.22	38	2118.5	2197	same
te120	10	0.45	103	3672.0	0.08	111	6974.8	1329	same
te120	20	0.65	87	8297.1	0.00	82	15509.9	921	same
tc160	5	0.58	173	2834.5	(-0.10)	43	4662.6	2083	2081
tc160	10	0.45	195	8159.7	(-0.30)	20	13957.6	1323	1319
tc160	20	1.03	25	19999.4	(-0.62)	197	37139.9	966	960
te160	5	0.46	137	4139.6	0.07	129	5954.8	2790	same
te160	10	0.18	108	10784.2	(-0.24)	83	17257.9	1650	1646
te160	20	0.18	166	27678.6	(-0.27)	57	45129.8	1101	1098

6. Concluding remarks

We first presented a local search heuristic for the capacitated minimum spanning tree problem, using a new path-based neighborhood structure and reduction techniques to speed up the search. Computational results on a set of standard benchmark problems showed that the proposed local search considerably improved the solutions found by a well known savings heuristic, leading to good trade-offs between solution quality and computation time.

We also proposed a GRASP heuristic using this neighborhood structure within the local search phase. One possible shortcoming of the standard memoryless GRASP metaheuristic being the independence of its iterations, we improved the local search strategy by incorporating short- and long-term memory features to it. A very short term memory is used to implement an extended local search going beyond the first local optimum, by the incorporation of some elements of tabu search. Long term memory information gathered from good solutions found along the search is used to implement a path-relinking procedure for intensification. Computational results obtained for benchmark problems illustrated the effectiveness of the proposed heuristic, which is competitive with the best heuristic in the literature in terms of solution quality. The GRASP heuristic using a memory-based local search strategy improved the best known solution for five out of the six largest benchmark problems.

Acknowledgements: The authors are grateful to L. Gouveia and P. Martins for kindly making available their test problems. Work of the first author was sponsored by CNPq in the framework of the doctorate scolarship grant number 200133/98-05. Work of the third author was sponsored by FAPERJ grant 150966/99 and by CNPq grants 302281/85-1, 202005/89-5, and 910062/99-4.

References

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.

R.M. Aiex, M.G.C. Resende, P.M. Pardalos, and G. Toraldo, "GRASP with path-relinking for the three-index assignment problem", Technical report, AT&T Labs-Research, 2000.

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

S.A. Canuto, M.G.C. Resende, and C.C. Ribeiro, "Local search with perturbations for the prize-collecting Steiner tree problem in graphs", *Networks* 38 (2001), 50–58.

K.M. Chandy and T. Lo, "The capacitated minimum spanning tree", *Networks* 3 (1973), 173–181.

K.M. Chandy and R.A. Russel, "The design of multipoint linkages in a teleprocessing tree network", *IEEE Transactions on Computers* 21 (1972), 1062–1066.

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

T.A. Feo and M.G.C. Resende, "Greedy randomized adaptive search procedures", *Journal of Global Optimization* 6 (1995), 109–133.

P. Festa and M.G.C. Resende, "GRASP: An annotated bibliography", in *Essays and Surveys in Metaheuristics* (C.C. Ribeiro and P. Hansen, eds.), pages 325–367, Kluwer, 2002.

B. Gavish, "Topological design of centralized computer networks: Formulations and algorithms", *Networks* 12 (1982), 355–377.

B. Gavish, "Formulations and algorithms for the capacitated minimal directed tree problem", *Journal of the ACM* 30 (1983), 118–132.

F. Glover, "Tabu search and adaptive memory programing – Advances, applications and challenges", in *Interfaces in Computer Science and Operations Research* (R.S. Barr, R.V. Helgason, and J.L. Kennington, eds.), pages 1–75, Kluwer, 1996.

F. Glover, "Multi-start and strategic oscillation methods – Principles to exploit adaptive memory", in *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research* (M. Laguna and J.L. Gonzáles-Velarde, eds.), pages 1–24, Kluwer, 2000.

F. Glover and M. Laguna, *Tabu Search*, Kluwer, 1997.

F. Glover, M. Laguna, and R. Martí, "Fundamentals of scatter search and path relinking", *Control and Cybernetics* 39 (2000), 653–684.

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

L. GOUVEIA AND P. MARTINS, "The capacitated minimum spanning tree problem: An experiment with a hop-indexed model", *Annals of Operations Research* 86 (1999), 271–294.

L. GOUVEIA AND P. MARTINS, "A hierarchy of hop-indexed models for the capacitated minimum spanning tree problem", *Networks* 35 (2000), 1–16.

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

J.B. KRUSKAL, "On the shortest spanning tree of a graph and the traveling salesman problem", *Proceedings of the American Mathematical Society* 7 (1956), 48–50.

M. LAGUNA AND R. MARTÍ, "GRASP and path relinking for 2-layer straight line crossing minimization", *INFORMS Journal on Computing* 11 (1999), 44–52.

K. MALIK AND G. YU, "A branch and bound algorithm for the capacitated minimum spanning tree problem", *Networks* 23 (1993), 525–532.

P. MARTINS, "Problema da árvore de suporte de custo mínimo com restrição de capacidade: Formulações com índice de nível", Ph.D. thesis, Departamento de Estatística e Investigação Operacional, Universidade de Lisboa, 1999.

P. MARTINS, personnal communication, 2001.

C. PAPADIMITRIOU, "The complexity of the capacitated tree problem", *Networks* 8 (1978), 217–230.

M.G.C. RESENDE AND C.C. RIBEIRO, "A GRASP with path-relinking for permanent virtual circuit routing", to appear in *Networks*, 2002.

M.G.C. RESENDE AND C.C. RIBEIRO, "GRASP", to appear in *State-of-the-Art Handbook of Metaheuristics* (F. Glover and G. Kochenberger, eds.), Kluwer, 2002.

C.C. RIBEIRO, E. UCHOA, AND R.F. WERNECK, "A hybrid GRASP with perturbations for the Steiner problem in graphs", *INFORMS Journal on Computing* 14 (2002), 228–246.

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.

P.M. THOMPSON AND J.B. ORLIN, "The theory of cyclic transfers", Working paper OR200-89, MIT, Operations Research Center, 1989.

P.M. THOMPSON AND H.N. PSARAFTIS, "Cyclic transfer algorithms for multi-vehicle routing and scheduling problems", *Operations Research* 41 (1993), 935–946.