An edge-based crossover for the capacitated minimum spanning tree problem

E. G. M. de Lacerda and Manoel Firmino de Medeiros Junior

***Index Terms*—genetic algorithms, spanning tree, capacitated minimum spanning tree problem**

***Abstract*—This work describes a Genetic Algorithm for the Capacitated Minimum Spanning Tree (CMST) problem that appears in Telecommunications. We suggest a new crossover operator, applied directly on the tree (phenotype) instead of on the chromosome. Without needing repairing techniques, this crossover is capable to produce feasible trees and presents excellent locality and heritability besides other properties that misses in many other tree representations. The new crossover proved to be effective when compared with the Genetic Algorithm of Raidl and Drexel (on benchmark data sets), which presented better performance than various classical methods of the literature. Another characteristic of the new crossover is its flexibility for adaptation in problems with constraints, as for instance, the Degree-Constrained Minimum Spanning Tree Problem.**

I. Introduction

The Capacitated Minimum Spanning Tree Problem (CMST) appears in the design of telecommunication networks [12]. Consider $G = (V, E)$ as being a connected and undirected graph, where $V = 1, 2, \ldots, n$ is the set of nodes of G and E is the set of edges. The special node $i = 1$ is named central node in which is root of the tree. The other nodes are called client nodes. Consider the following components of the problem:

- For each client node $i \in V \setminus \{1\}$, there is an associated demand $d_i > 0$, representing the flow from the central node to the node i;
- For an edge $(i, j) \in E$, there is a cost $c_{i,j} > 0$ associated;
- For all edges $(i, j) \in E$, there is a maximum capacity $K > 0$ that limits the flow on (i, j).

The problem is to find a spanning tree T on G that minimizes the cost:

$$C(T) = \sum_{(i,j)\in T} c_{i,j}$$

satisfying the constraint:

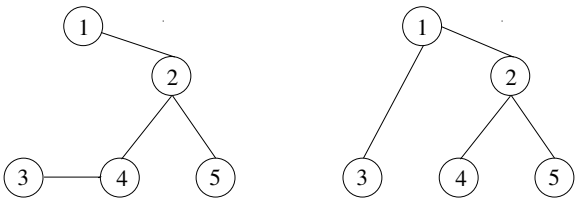
$$d_j + \sum_{k\in \mathrm{Sub}(j)} d_k \leq K \qquad \forall (i, j) \in T$$

where $\mathrm{Sub}(j)$ represents the set of all of the nodes $k \neq j$ that contains the node j in the path that takes them to the central node.

For example consider a graph with nodes $i = 1, \ldots, 5$ and demands $d_2 = 1$, $d_3 = 4$, $d_4 = 2$ and $d_5 = 1$. Consider the maximum capacity $K = 5$. The Figure 1(a) displays one unfeasible tree on this graph. The required flow on the edge $(2, 4)$ (which is equal to $d_3 + d_4 = 4 + 2 = 6$) exceeds the maximum capacity. Therefore, this tree does not satisfy the capacity constraints.

E. G. M. de Lacerda, UFRN - CT - DCA, 59072-970 - Natal - RN - Brazil, Fone: 0(xx)(84)215-3771 R. 201, Fax: 0(xx)(84)215-3738, e-mail:estefane@dca.ufrn.br

Manoel Firmino de Medeiros Junior, UFRN - CT - DCA, 59072-970 - Natal - RN - Brazil, Fone: 0(xx)(84)215-3771 R. 201, Fax: 0(xx)(84)215-3738, e-mail:firmino@dca.ufrn.br



(a) Infeasible tree (b) Feasible tree

Figure 1. Examples of trees for CMST problem

The tree of Figure 1(b) is feasible, satisfying all of the capacity constraints.

II. Previous Works

Amberg et al [4] presents a survey of several algorithms for the problem of CMST in which it involves heuristics and metaheuristics such as Tabu Search and Simulated Annealing. Sharaiha et al [29] and Ahuja et al [2], [3] propose efficient Tabu Search algorithms for this problem. Metaheuristics like Ant Colony [26], GRASP [10] and adaptive reasoning technique [20] were also proposed to solve the CMST problem. The CMST problem was proved to be NP-complete by Papadimitriou [19].

In spite of the literature of Genetic Algorithms disposal several works for problems related with spanning trees, few of them were dedicated to the specific CMST problem (for example, the Raidl and Drexel's GA [25]).

To proceed, will be presented several representations of trees. Throughout the text, the algorithms use the pseudocode notation from [9]. We will begin describing relevant criteria for a good tree representation choice.

III. Desirable Properties of Tree Representations

The performance of the genetic algorithm depends crucially on the representation and on how the genetic operators interact with this representation. Researchers [16], in general, agree that the following properties are desirable in a good representation:

- *Space:* chromosomes should not occupy great amount of memory;
- *Time:* the complexity of time to decode, to apply crossover and mutation should be small;
- *Feasibility:* the population and the chromosomes generated by crossover and mutation should represent only feasible solutions;
- *Coverage:* the representation should be capable to represent all the feasible trees;

- *Locality:* chromosomes that suffer small mutations should represent solutions (trees) similar to the original chromosome. In other words, similar chromosomes should represent similar solutions;
- *Constraints:* the representation should allow the incorporation of constraints. For instance, constraints that limits the degree of the nodes of the graph;
- *Heritability:* chromosomes should be formed through the combination of substructures of the parental solutions. For instance, most of the edges of a tree should belong to its parents;
- *Hibrids:* the representation should allow the incorporation of heuristics (as for instance, heuristic that favor the low cost edges);
- *Nonredundancy:* the mapping between genotypes and trees must be made one-to-one. If many-to-one mapping occurs, the number of genotypes exceeds the number of trees. In this case, the GA wastes time in searching, because one or more trees may be duplicated in the genotype space.

IV. Tree Representations

A. Characteristic Vectors

A characteristic vector is a bit string of length equal to m (number of edges of G). Each bit of the string corresponds to an edge of the graph G and indicates whether the edge of G is or not present in the solution represented by the string. In this representation, not every string represents a spanning tree.

In a complete graph, there exists $m = n(n-1)/2$ edges, and therefore, the string can represent $2^{n(n-1)/2}$ solutions. It can be proved that a complete graph has n^{n-2} different spanning trees [6]. So the fraction of feasible solutions (spanning trees) in relation to the number of solutions representing by the string is:

$$\frac{n^{n-2}}{2^{n(n-1)/2}}, \tag{1}$$

The value of the Equation (1) is small for big n and it becomes infinitesimally small as n grows. Because of this, the characteristic vector has poor feasibility once the most of its solutions are unfeasible, in spite of having good locality and heritability. Besides, the traditional crossover between two feasible solutions, in general, generates an unfeasible solution.

B. Predecessor

In this representation, it is designated arbitrarily a root node. The chromosome is a array P. P stores, in position i, the predecessor node of the node i of the tree, in other words, if $P[i] = j$ then j is the first node of the path that goes from i to the root. The value $P[i]$ can assume any node, that is, $P[i] \in \{1, 2, 3, \ldots, n\}$. Because a tree has $n-1$ edges, then the length of the chromosome P is $n-1$. The tree of the Figure 2 has the following representation (assuming that the root is the node $i = 1$):

$$P = (1, 5, 7, 2, 7, 5, 7)$$

The chromosome P is capable of representing $n-1$ different solutions. Therefore, the fraction of feasible solutions in relation to the number of solutions represented by P it is given by

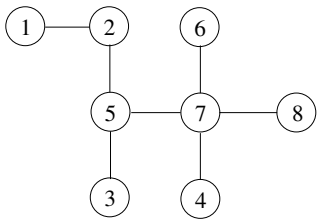


Figure 2. Tree represented by vector $P = (1, 5, 7, 2, 7, 5, 7)$ by using the predecessor representation. The same tree can be represented by a smaller vector $P = (2, 5, 5, 7, 7, 7)$ by using Prüfer Numbers.

$$\frac{n^{n-2}}{n^{n-1}} = \frac{1}{n}$$

in which it is a great improvement in relation to the characteristic vector. But there is still a lot of unfeasible solutions. Besides, the crossover can also generate unfeasible solutions. The Predecessor encoding was used in [1], [8], [5], [7].

C. Prüfer Numbers

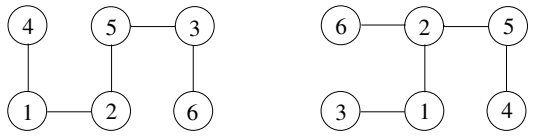
The Cayley's formula [6] shows that there are n^{n-2} trees in a complete graph of n nodes. Prüfer [22] presented a proof of Cayley's formula by inventing one string (the Prüfer number) with $n-2$ digits that determines uniquely one tree.

Unlike Predecessor encoding and the Characteristic Vector, the chromosomes codified with the Prüfer number only generate feasible solutions. It happens because the mapping between a tree T and its corresponding Prüfer number $P(T)$ is one-to-one. We will just show the conversion of one tree into one Prüfer number. Details of the conversion in the opposite direction are in [30]. This algorithm generates the Prüfer number $P(T)$ starting from the tree T:

1. Start $P(T)$ with $n-2$ empty positions of digits;
2. Let i be the leaf node of lowest number in T. Consider the node j as being the predecessor of i. Then j will occupy the leftmost empty position of $P(T)$;
3. Remove i and the edge (i, j) of the tree T;
4. If there are only two nodes in the tree, stop. Otherwise, go to the step 2.

For instance, the Prüfer number for the tree of the Figure 2 is determined as follows. As their leaves are $\{1, 3, 6, 4, 8\}$ then the lowest numbered leaf is 1 and its predecessor is 2, which it occupies the empty leftmost position of $P(T) = (2, \square, \square, \square, \square, \square)$. In the step 3, it occurs the removal of node 1. Repeating the step 2, the leaves of T are now $\{2, 3, 6, 4, 8\}$ whose lowest numbered leaf is 2 (with predecessor 5). Then, $P(T) = (2, 5, \square, \square, \square, \square)$. In the step 3, it occurs the removal of node 2. Repeating the step 2, the leaves of T are now {3, 6, 4, 8} whose lowest numbered leaf is 3 (with predecessor 5). Then $P(T) = (2, 5, 5, \square, \square, \square)$. Continuing to repeat the steps 2 and 3 until remaining only two leaves in T, we will obtain $P(T) = (2, 5, 5, 7, 7, 7)$.

Researchers [13] pointed out that the Prüfer number has poor locality and heritability. A poor locality means that similar chromosomes generate very different solutions. For instance, the Figure 3(a) illustrates the Prüfer number $P_1 = (1, 2, 5, 3)$



(a) $P_1 = (1, 2, 5, 3)$ (b) $P_2 = (1, 2, 5, 2)$

Figure 3. The locality problem in Prüfer numbers

KRUSCALRST(V, E)
1 $T \leftarrow \emptyset$, $A \leftarrow E$
2 **while** $|T| < |V| - 1$
3 **do** choose an edge $(i, j) \in A$ at random
4 $A \leftarrow A - \{(i, j)\}$
5 **if** i and j are not yet connected in T
6 **then** $T \leftarrow T \cup \{(i, j)\}$
7 **return** T

Figure 4. The KruscalRST Algorithm

and its associated tree. In the Figure 3(b), the Prüfer number suffer a small mutation (in the last position) and its associated tree becomes very different from the original tree. Futhermore, Prüfer number do not also allow the incorporation of constraints in which is required in many spanning problems.

There are many other one-to-one mappings [21] among strings of $n-2$ digits and trees. Some exhibits better locality than the Prüfer number as, for instance, the Blob Code. This explains the Blob Code's better performance in comparison to the Prüfer number mentioned in [13]. In spite of its proven inferiority, the Prüfer number have been used in several problems [30], [11].

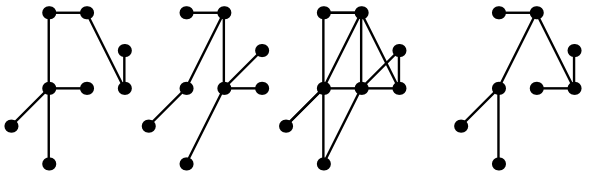
D. Edge-Sets

The code Edge-sets [23] is a direct tree representation because the chromosome is simply the set of edges of the tree.

Each chromosome from the initial population is generated through a slightly modified version of the classic Kruscal algorithm [9]. This version is denominated KruscalRST [9] (Kruscal Random Spanning Tree). The difference between the Kruscal algorithm and the KruscalRST is that the first chooses an edge in agreement with its cost and the second chooses an edge randomly. The algorithm KruscalRST is shown in the Figure 4.

The crossover operator is denominated KruscalRST crossover. It generates a child starting from two parents, T_1 and T_2, applying the KruscalRST algorithm in the graph formed by the union of the edges T_1 and T_2, i.e., $G' = (V, T_1 \cup T_2)$. See an example in the Figure 5. This operator has high heritability once the children are generated using just the parental edges.

This operator can be adapted to deal with spanning tree problems with restrictions as, for instance, the Degree-Constrained Minimum Spanning tree (DegMST). In the DegMST problem the maximum number of adjacent edges to any node is limited. However, Raidl and Julstrom [23] showed that, in this case, the KruscalRST crossover can generate unfeasible trees. To overcome this problem, the crossover should use edges that do not belong to the parents to turn feasible the child (losing, however, a little of heritability).



(a) T_1 (parent) (b) T_2 (parent) (c) $T_1 \cup T_2$ (d) KruscalRST

Figure 5. Example of KruscalRST Crossover

TABLE I
SOME PROPERTIES OF TREE REPRESENTATIONS

Representation	Feasib.	Local.	Herit.	Constraints
Char. Vector	worst	high	high	avg.
Predecessor	poor	high	high	avg.
Prüfer Numbers	poor	low	low	poor
Blob Code	yes	avg.	avg.	poor
link-&-node biased	yes	avg.	avg.	good
Network random Keys	yes	avg.	avg.	good
Edge Sets	yes	high	high	good

In [23] is described several variants of this crossover and heuristics for improving its performance. In [17], a different crossover is proposed for this same representation.

E. Other Representations

Several other representations has been used in the literature. Knowles and Corne [15] described an efficient representation for the DegMST problem. Others examples are Link-&-Node Biased [18], Blob Code [14], and Network Random Keys [28].

In Table I is shown some properties of tree representations (see [23] for details). The properties also depend on the genetic operators. For example, the Predecessor representation has poor feasibility if it is used with traditional operators. However if it is used with specialized operators such as Edge-crossover [25] and Pred-Crossover (see Section V) then it will have high feasibility.

V. A GENETIC ALGORITHM FOR THE CMST PROBLEM

In this section, a new crossover, called Pred-Crossover, will be described. The tree representation is the Predecessor one whose root is the central node ($i = 1$). A chromosome P is an array with elements $P[2], P[3], \ldots, P[n]$ where $P[i]$ is the predecessor of the client node i (as seen in Section IV).

The algorithm to create the initial population (with feasible trees) and the mutation operator (the edge-mutation) were taken from Raidl and Drexel's GA [25]. The Pred-Crossover creates

a child by swapping parental edges (on the condition that this is possible). Because the crossover is performed on phenotype space, it is not limited to the Predecessor representation, that is, it can be used in other representations too. Edge-mutation and Pred-Crossover are described as follows.

A. Edge-Mutation

The Edge-Mutation [25] replaces a predecessor of the node i with a random predecessor node (as long as the resulting tree is feasible).

The Edge-Mutation algorithm is described in Figure 6. In line 2, the algorithm removes the edge $(P[i], i)$ from the tree represented in P by assigning an invalid value to the predecessor of node i. The consequence of this is the tree divided into two unconnected components. One of the components is connected to root (central) node. The other component is connected to node i. The function $GetNodesConnectedToRoot(P)$ (line 3) gets all nodes connected to the root node. This can be done using depth first-search (including the node root). The remaining lines of the algorithm (lines 5 to 10) are a loop to select a random node from S until it finds a feasible predecessor node j for node i. In line 6, the function $random(S)$ selects a random node from the set S.

```
EDGEMUTATION(P, i)
 1  j ← P[i]
 2  P[i] ← NULL
 3  S ← GetNodesConnectedToRoot(P)
 4  S ← S − {j}
 5  while S ≠ ∅
 6        do j ← random(S)
 7           P[i] = j
 8           if isFeasible(P)
 9             then return P
10             else  S ← S − {j}
```

Figure 6. The Edge-Mutation

As a **local heuristic**, the low-cost edges are favored when selecting a random node from the set S (line 6). This is done by selecting first the predecessor nodes of the $n/8$ cheapest edges to node i.

B. Pred-Crossover

The Pred-Crossover takes two parents P and Q and builds a set S with all client nodes $i \in \{2, \ldots, n\}$ such that $P[i] \neq Q[i]$. Next it selects a random client node $a \in S$ and swaps $P[a]$ for $Q[a]$ (as long as the resulting tree is feasible). If it is possible, this operation is repeated until $l = |S|/2$ valid swaps have been done. If only $k < l$ swaps were done, build a set B with all nodes $i \in S$ such that $P[i]$ and $Q[i]$ were not swapped before. Next $l - k$ mutations are carried out in random nodes $a \in B$. The Pred-Crossover algorithm is described in Figure 7. Given two parents P and Q, two children $C1$ and $C2$ are created as follows:

$C1 \leftarrow PredCrossover(P, Q)$
$C2 \leftarrow PredCrossover(Q, P)$

The additional mutations in the end of the Pred-Crossover are optional (lines 18 to 21). However, this method increased the diversity of the GA and improved its performance in several benchmark CMST problems.

```
PREDCROSSOVER(P, Q)
 1  S = {i : P[i] ≠ Q[i]}
 2  l ← |S|/2, k ← 0, B ← ∅
 3
 4  ▷ Carry out l swaps if it is possible
 5  while S ≠ ∅ and k < l
 6        do a ← random(S)
 7           swap(P[a], Q[a])
 8           if isFeasible(P)
 9             then S ← (S ∪ B) − {a}
10                  B ← ∅
11                  k ← k + 1
12             else  ▷ Undo the invalid swap
13                  swap(P[a], Q[a])
14                  S ← S − {a}
15                  B ← B ∪ {a}
16
17  ▷ Carry out l − k mutations in P
18  for i ← k to l
19        do a ← random(B)
20           P ← EdgeMutation(P, a)
21           B ← B − {a}
22
23  return P
```

Figure 7. The Pred-Crossover

As a **local heuristic**, the low-cost edges are favored when selecting a random node from the set S (line 6). This is done by a tournament that works as the tournament selection. It uses a group with $n/4$ nodes from the set S. Next the tournament selects a node a from the group such that the edge $(Q[a], a)$ is the cheapest edge.

C. Constraints

The constraints are handled by the subroutine $isFeasible()$ in line 8 (Figure 7). It searches for violating of capacity constraints and for cycles in the graph. In order to check violation of capacity constraints efficiently can be useful to store, in each node i, the current flow required in the edge ($P[i]$,i).

It is worth noting that the subroutine $isFeasible()$ is independent one of the Pred-Crossover algorithm. So $isFeasible()$ can be modified to deal with other types of constraints. Because of this the Pred-Crossover can be applied to other spanning tree problems such as the Degree-constrained minimum spanning tree (DegMST) problem [30] [16] and the Bounded-Diameter Minimum Spanning Tree Problem (BDMST) [24].

For example, the DegMST problem has the following constraint: the maximum number of edges adjacent to any node is limited. If Pred-Crossover is applied to DegMST then the $isFeasible()$ subroutine is modified to check the number of edges adjacent to a node (instead of capacity constraints).

TABLE II
COMPARING THE PROPOSED GA WITH THE RAIDL AND DREXEL'S GA ALGORITHM

				GA with Pred-Crossover			Raidl and Drexel's GA			
Problem	K	C_{opt}		Best	Avg	σ	Best	Avg	σ	t-test (α)
tc40-1	5	586	opt	586	587.6	1.84	586	587.0	1.05	0.185%
tc40-2	5	578	lb	578	580.0	2.94	579	579.2	0.63	0.200%
tc40-3	5	577	opt	577	577.2	0.63	577	577.0	0.00	0.158%
tc40-4	5	617	opt	617	619.4	2.55	617	617.1	0.32	0.002%
tc40-5	5	600	lb	603	603.0	0.00	602	604.5	1.08	0.000%
tc40-1	10	498	opt	498	498.0	0.00	498	498.0	0.00	0.500%
tc40-2	10	490	opt	490	491.1	3.48	490	490.4	0.84	0.268%
tc40-3	10	500	opt	500	500.0	0.00	500	501.8	2.90	0.025%
tc40-4	10	512	opt	512	512.0	0.00	512	512.4	0.70	0.035%
tc40-5	10	504	opt	504	504.0	0.00	504	504.0	0.00	0.500%
te40-1	5	830	lb	834	838.8	5.07	830	833.9	2.18	0.002%
te40-2	5	792	lb	793	798.1	4.53	792	797.4	8.11	0.406%
te40-3	5	797	lb	800	806.0	4.14	801	806.0	3.77	0.500%
te40-4	5	814	lb	815	820.7	4.24	814	822.0	4.81	0.261%
te40-5	5	784	lb	784	788.6	3.03	784	784.4	1.26	0.000%
te40-1	10	596	lb	596	601.7	5.54	596	599.4	4.99	0.165%
te40-2	10	573	lb	577	578.8	1.55	581	581.0	0.00	0.000%
te40-3	10	568	lb	568	571.0	2.54	568	569.1	1.66	0.024%
te40-4	10	596	lb	596	597.2	1.03	596	597.6	0.84	0.171%
te40-5	10	572	opt	572	573.4	1.26	572	575.2	1.93	0.007%

VI. EXPERIMENTAL RESULTS

The experimental results were obtained on a benchmark data set from the OR-Library of J. E. Beasley[1]. This data set has been used in several works [4]. All client nodes have unit demands ($d_i = 1$). There are two categories of problem instances named *tc* and *te*. In the category *tc*, the root node is centered among the client nodes whereas in the category *te* the root node is located near the border of the convex hull of the client node set. In both categories, there are five cost matrices for $n = 40$ nodes and two different values for edge capacity K. So there are $2 \times 5 \times 2 = 20$ problem instances.

The GA uses the so-called steady-state replacement. In this scheme, only one child is created in each generation. Such a child replaces the worst individual of the population. In order to avoid loss of diversity, the child is discarted if it is already in the population.

The parents are selected by the probabilistic binary tournament selection in which the better parent wins the tournament with probability equal to $p > 0.5$. If $p = 1.0$, it is identical to the traditional binary tournament selection.

The remaining parameters of GA are described as follows.

- The population size is equal to 500;
- The crossover is always carried out;
- The mutation is carried out in each edge with mutation rate equal to 0.05;
- The population initial is created using the same parameters as Raidl and Drexel's GA [25];
- The probabilistic binary tournament selection uses $p = 0.6$ (for low selection pressure);
- Stopping criterion: if no progress had been made in 20000 individual evaluations then the GA stop.

Table II shows the results for tc and te problem instances. 10 runs were performed per instance. The column C_{opt} shows optimum objective values or lower bounds according to [27]. Other columns show the best values, the average values and the standard deviations for each GA. The last column lists the error probabilities in t-test of hypotheses that differences exist among the two GAs. Hence we may conclude with high confidence the two GA are similar in most of the problem instances.

VII. CONCLUSIONS AND FUTHER WORK

This work described a crossover (the Pred-Crossover) using the Predecessor encoding. This crossover was designed to avoid the drawbacks found in other representations that do not allow the incorporation of constraints or that have lack of feasibility, locality, or heritability. The Pred-Crossover presented good performance on a benchmark data set. Despite the result obtained was similar to the Raidl and Drexel's GA, the Pred-Crossover is more flexible in sense that it can be applied to other different spanning tree problems (whereas the Raidl and Drexel's GA is specific for the CMST problem). It occurs because the constraint checking of the Pred-Crossover algorithm is an independent subroutine and because of this it can be adapted for different types of constraints.

It is worth mentioning the experiments was dramatically improved by local heuristics for mutation, crossover and ini-

[1] `http://mscmga.msc.ic.ac.uk/info.html`

tial population. Hence future work should explore other local heuristics for the CMST problem. Future work should also apply the Pred-Crossover in other categories of CMST problems and in other constrained problems such as the DegMST and the BDMST problems.

ACKNOWLEDGMENTS

The authors would like to thank CNPq and FAPERN for financial support.

REFERENCES

[1] F. N. Abuali, R. L. Wainwright, and D. A. Schoenefeld. Determinant factorization: A new encoding scheme for spanning trees applied to the probabilistic minimum spanning tree problem. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 470–477, San Mateo, CA, 1995. Morgan Kaufmann Publishers.

[2] Ravindra K. Ahuja, James B. Orlin, and Dushyant Sharma. Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. *Math. Program*, 91:71–97, 2001.

[3] Ravindra K. Ahuja, James B. Orlin, and Dushyant Sharma. A composite very large-scale neighborhood structure for the capacitated minimum spanning tree problem. *Operations Research Letters*, 31:185–194, 2003.

[4] A. Amberg, W. Domschke, and S. Vob. Capacitated minimum spanning trees: Algorithms using intelligent search. *Combinatorial Optimization: Theory and Practice*, 1(1):9–39, 1996.

[5] Les Berry, Bruce Murtagh, Steve Sugden, and Graham McMahon. Application of a genetic-based algorithm for optimal design of tree-structured communication networks. In *Proceedings of the Regional Teletraffic Engineering Conference of the International Teletraffic Congress*, pages 361–370, South Africa, 1995.

[6] A. Cayley. A theorem on trees. *Quarterly Journal of Mathematics*, 23:376–378, 1889.

[7] Hsinghua Chou, G. Premkumar, and Chao-Hsien Chu. Genetic algorithms for communications network design - an empirical study of the factors that influence performance. *IEEE Transactions on Evolutionary Computation*, 5(3):236–249, 2001.

[8] C.-H. Chu, G. Premkumar, C. Chou, and J. Sun. Dynamic degree constrained network design: A genetic algorithm approach. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, pages 141–148. Morgan Kaufmann, 1999.

[9] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001.

[10] M.C. de Souza, C. Duhamel, and C.C. Ribeiro. A GRASP heuristic for the capacitated minimum spanning tree problem using a memory-based local search strategy. In M.G.C. Resende and J.P. de Sousa, editors, *Metaheuristics: Computer decision-making*, pages 627–658. Kluwer Academic Publishers, 2003.

[11] M. L. Gargano, W. Edelson, and O. Koval. A genetic algorithm with feasible search space for minimal spanning trees with time-dependent edge costs. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1998, Proceedings of the Third Annual Conference*, page 495. Morgan Kaufmann, 1998.

[12] B. Gavish. Topological design of telecommunication networks - local access design methods. *Annals of Operations Research*, 33:17–71, 1991.

[13] Jens Gottlieb, Bryant A. Julstrom, Franz Rothlauf, and Güunther R. Raidl. Prüfer numbers: A poor representation of spanning trees for evolutionary search. In Lee Spector, Erik Goodman, Annie Wu, W. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max Garzon, and Edmund Burke, editors, *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, pages 343–350. Morgan Kaufmann, 2000.

[14] Bryant A. Julstrom. The blob code: A better string coding of spanning trees for evolutionary search. In Robert Heckendorn, editor, *2001 Genetic and Evolutionary Computation Conference Workshop Program*, pages 256–261, San Francisco, CA, 2001.

[15] J. Knowles and D. Corne. A new evolutionary approach to the degree constrained minimum spanning tree problem. *IEEE Transactions on Evolutionary Computation*, 4(2):125–134, 2000.

[16] Mohan Krishnamoorthy and Andreas T. Ernst. Comparison of algorithms for the degree constrained minimum spanning tree. *Journal of Heuristics*, 7:587–611, 2001.

[17] Y. Li and Y. Bouchebaba. A new genetic algorithm for the optimal communication spanning tree problem. In Cyril Fonlupt, Jin-Kao Hao, Evelyne Lutton, Edmund Ronald, and Marc Schoenauer, editors, *Proceedings of Artificial Evolution: Fourth European Conference, vol. 1829 of LNCS*, pages 162–173. Springer, 1999.

[18] C. C. Palmer and A. Kershenbaum. Representing trees in genetic algorithms. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 379–384. IEEE Press, 1994.

[19] C. H. Papadimitriou. The complexity of the capacitated tree problem. *Networks*, 8:217–230, 1978.

[20] R. Patterson, E. Rolland, and H. Pirkul. A memory adaptive reasoning technique for solving the capacitated minimum spanning tree problem. *Journal of Heuristics*, 5(2):159–180, 1999.

[21] Sally Picciotto. *How to Encode a Tree*. PhD thesis, University of California, San Diego, 1999.

[22] H. Prüfer. Neuer beweis eines satzes über permutationen. *Archiv für Mathematik und Physik*, 27:141–144, 1918.

[23] G. R. Raidl and B. A. Julstrom. Edge-Sets: An effective evolutionary coding of spanning trees. Technical Report TR-186-1-01-01, Technisch Universität Wien, Institut für Computergraphik und Algorithmen, Wien, Austria, 2002.

[24] G.R. Raidl and B.A. Julstrom. Greedy heuristics and an evolutionary algorithm for the bounded-diameter minimum spanning tree problem. In G.Lamont etal editors, editor, *Proceedings of the 2003 ACM Symposium on Applied Computing*, pages 747–752. ACM Press, 2003.

[25] Günther R. Raidl and Christina Drexel. A predecessor coding in an evolutionary algorithm for the capacitated minimum spanning tree problem. In Chris Armstrong, editor, *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, pages 309–316, Las Vegas, NV, 2000.

[26] M. Reimann and M. Laumanns. A hybrid ACO algorithm for the capacitated minimum spanning tree problem. In Christian Blum, Andrea Roli, and Michael Sampels, editors, *Proceedings of First International Workshop on Hybrid Metaheuristics (HM2004)*, 2004.

[27] E. Rolland, R. Patterson, and H. Pirkul. Memory adaptive reasoning and greedy assignment techniques for the capacitated minimum spanning tree problem. In *Advances in Metaheuristics*, pages 485–497. Kluwer Academic, 1998.

[28] Franz Rothlauf, David Goldberg, and Armin Heinzl. Network random keys - a tree network representation scheme for genetic and evolutionary algorithms. *Evolutionary Computation*, 10(1):75–97, 2002.

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

[30] G. Zhou and M. Gen. Approach to degree-constrained minimum spanning tree problem using genetic algorithm. *Engineering Design e Automation*, 3(2):157–165, 1997.