A Predecessor Coding in an Evolutionary Algorithm for the Capacitated Minimum Spanning Tree Problem

Günther R. Raidl, Christina Drexel
Institute of Computer Graphics
Vienna University of Technology
Favoritenstraße 9–11/1861, 1040 Vienna, Austria
raidl@apm.tuwien.ac.at, e9225887@student.tuwien.ac.at

Abstract

This article presents an evolutionary algorithm (EA) for the capacitated minimum spanning tree problem occurring in telecommunication applications. The EA encodes a solution by a predecessor vector indicating for each node the preceding node at the path to the given central root node. Initialization, crossover, and mutation operators were specifically designed to provide strong locality and to enable an effective search in the space of feasible solutions only. Furthermore, local heuristics are applied to promote the inclusion of low-cost links. Empirical results on a set of standard test problems indicate that the EA performs better than two other heuristic techniques.

1 Introduction

The problem of finding a *capacitated minimum spanning tree* (CMST) plays an important role in the design of telecommunication networks [Gavish, 1991]. In this problem, a central server and a set of $n \geq 1$ client machines are given. Each client i ($i = 1, \ldots, n$) has an associated demand $d_i > 0$ representing the network traffic that is supposed to flow between the server and the client. The objective is to build a cheapest possible tree-network connecting all nodes, i.e. the server and the clients, which carries the specified demands. Between any pair of nodes, a link can be established for a given cost $c_{i,j} > 0$ ($i, j = 0, \ldots, n$, $i \neq j$; node 0 denotes the server). Note that $c_{i,j} = c_{j,i}$. Besides its cost, each potential link has associated a maximum capacity $m_{i,j} > 0$ restricting the total network traffic that may go over this link.

More formally, we want to identify a spanning tree T over all nodes $i = 0, \ldots, n$ with minimum total costs

$$C = \sum_{(i,j) \in T} c_{i,j} \tag{1}$$

that fulfills the additional capacity-constraints

$$d_j + \sum_{k \in Sub(j)} d_k \leq m_{i,j} \quad \forall (i,j) \in T\,, \tag{2}$$

where $Sub(j)$ denotes the set of all nodes $k \neq 0$ and $k \neq j$ having node j in their communication paths to the server. In other words, when we look at the network as a directed tree (more precisely an outgoing arborescence) with the server node 0 being the root, $Sub(j)$ denotes the set of all subnodes of node j.

Figure 1 shows an example of a feasible solution. All client nodes are supposed to have unit demand ($d_i = 1$), and each link's maximum capacity $m_{i,j}$ is fixed to 3. The link costs $c_{i,j}$ are printed, giving a total of $C = 176$. If e.g. node 9 would be reconnected to node 3, the tree would become infeasible since the needed capacity on link $(0, 1)$ would become 4 and therefore larger than $m_{0,1} = 3$.

In contrast to the unconstrained minimum spanning tree problem, which can be solved efficiently by well-known algorithms such as [Prim, 1957], the CMST problem has been shown to be $\mathcal{NP}$-complete by Papadimitriou [1978]; see also [Garey and Johnson, 1979].

The next section gives a short overview on previous approaches for the CMST and related problems. In sections 3 and 4 a new evolutionary algorithm (EA) is proposed. Its predecessor encoding and specialized initialization and variation operators, which also include local heuristics, are described in detail. Although effective evolutionary techniques have already been developed for related spanning tree problems, it is to our knowledge the first time that an EA is applied to the

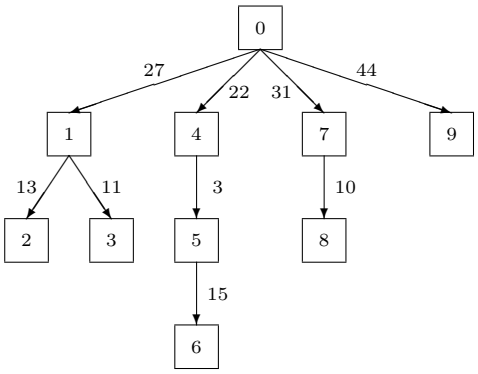

Figure 1: An exemplary capacitated spanning tree; link costs $c_{i,j}$ are shown, $d_i = 1$, $m_{i,j} = 3$.

CMST problem. Section 5 presents empirical results on a standard test problem set indicating that the new approach is effective. Finally, conclusions are drawn and remarks on further research are given in section 6.

2 Prior work

One of the first and best-known heuristics for the CMST problem has been proposed by Esau and Williams [1966]. This procedure, which we denote here as EW heuristic, starts from the root node as initial tree and iteratively adds links in a greedy fashion based on a modified cost structure without violating capacity constraints, until a complete spanning tree is found. In this process, the original costs of each link are reduced by the minimum cost of the adjacent nodes' direct links to the root node.

Amberg et al. [1996] give a survey on several algorithms published for the CMST problem during the last decades and present a new heuristic approach that differs from others in that it is based on partitioning nodes into subsets instead of focusing on the selection of links when generating or transforming solutions. To overcome local optimality, simulated annealing and tabu search were investigated. Another effective tabu search approach for the CMST has been proposed by Sharaiha et al. [1997]. The neighborhood structure in this approach is based on the exchange of a single node or a set of nodes between two subtrees. Ahuja et al. [1998] extended this neighborhood structure by also allowing exchanges that involve multiple subtrees.

An exact method based on a branch-and-bound algorithm with column generation and the utilization of cutting planes to strengthen the formulation is proposed in [Chang and Vance, 1998]. [Hall, 1996] presented another cutting plane algorithm for the CMST problem. Due to the complex nature of the problem, the applicability of such exact techniques is limited to relatively small problem instances or special problem subclasses.

Recently, Patterson et al. [1999] proposed a hybrid memory *adaptive reasoning technique* (ART) for solving the CMST problem, see also [Rolland et al., 1998]. In this approach, the classic EW heuristic is iteratively applied to the problem which is augmented with additional non-redundant constraints determined via an adaptive memory. Empirical results indicate that the approach performs on par with the best other heuristic techniques, while expending only a small amount of computational effort.

Although EAs are known to perform very well on several other spanning tree related problems, we are not aware of any published EA for the CMST problem up to now. EAs for related spanning tree problems differ mainly in the way how solutions are encoded, see [Palmer and Kershenbaum, 1994] and [Raidl, 2000] for overviews.

Primitive encodings such as bit strings indicating directly which edges are to be included in a solution have inherent problems with the huge amount of infeasible solutions that are generated by an EA using traditional variation operators [Piggott and Suraweera, 1995].

An often applied coding for spanning trees is based on Prüfer's proof of Cayley's formula, which identifies the number of distinct spanning trees on a complete graph with k vertices as k^{k-2} [Cayley, 1889]. The proof establishes a one-to-one correspondence between spanning trees and strings of length $k-2$ over an alphabet of k symbols by describing fast algorithms that derive a tree from its string and vice versa [Palmer and Kershenbaum, 1994].

Zhou and Gen [1997] proposed an EA based on Prüfer numbers for the *degree-constrained minimum spanning tree* (DegMST) problem in which the maximum number of edges adjacent to any node is limited. Conventional uniform crossover and mutation by randomly modifying a symbol were applied. A similar EA for bicriteria topological network design is described in [Kim and Gen, 1999]. Two more elaborated variants of Prüfer-coded EAs for the DegMST problem are proposed in [Krishnamoorthy et al., 1999]. Abuali et al. [1994] used the Prüfer coding in a GA for the probabilistic minimum spanning tree problem. Operators included circular left shift, swap of two random positions, and random modification of a symbol.

However, several researchers have pointed out that the Prüfer number encoding is problematic due to its weak

locality [Palmer and Kershenbaum, 1994, Knowles and Corne, 2000, Raidl and Julstrom, 2000]. Rothlauf and Goldberg [1999] studied the locality properties of the Prüfer encoding in detail. Patterns of symbols do not represent consistent substructures of spanning trees. Usually applied variation operators change represented trees radically. Often, a generated offspring does not have many edges in common with its parent(s).

For the DegMST problem, Knowles and Corne [2000] described a more effective EA in which chromosomes are sequences of integer values that influence the order in which a modified version of Prim's MST algorithm [Prim, 1957] connects vertices to the growing spanning tree.

A more general decoder-based technique is the weight-coded approach, which was first applied by Palmer and Kershenbaum [1994] to the optimum communication spanning tree problem and later significantly improved and applied to the DegMST problem by Raidl and Julstrom [2000]: A solution is encoded as a vector of usually real-valued weights. For obtaining the represented phenotypic spanning tree, the weights are used to bias parameters of the problem such as the link costs; in that way, a temporary modified problem is generated. A problem specific decoding heuristic is then used to derive the phenotypic solution, which is finally evaluated using the original problem data. With this approach significantly better results have been achieved than when using Prüfer number encoding.

Finally, there is a category of EAs which use a relatively simple and direct encoding and specialized initialization and variation operators to always generate only feasible solutions. Li and Bouchebaba [1999] describe such an approach for the optimum communication spanning tree problem which performs slightly better than the weight-coded approach of Palmer and Kershenbaum [1994]. In [Raidl, 2000] an EA using an edge-list encoding with problem-dependent initialization, recombination, and mutation methods is presented for the DegMST problem. The operators are designed in such a way that they are computationally relatively inexpensive. Nevertheless, they produce only feasible solutions, provide strong locality, and include effective local heuristics for the particular target problem. This approach performs even better than the previous weight-coded approach [Raidl and Julstrom, 2000] and especially scales well to large problem instances.

The many published EAs for spanning tree problems suggest that an approach which uses an encoding that directly reflects the spanning tree but includes problem-dependent, specialized initialization and variation operators may be the best choice. In the next sections, we describe an EA that follows these ideas and exploits the very specific properties of the CMST problem.

3 Predecessor Encoding

In contrast to the DegMST and optimum communication spanning tree problems, there exists a designated root node – the central server – in the CMST problem. We can therefore see a feasible solution as an outgoing arborescence, i.e. each link is interpreted as a directed edge in such a way that there exists a path from the root to each other node. Each client node has now exactly one preceding node, and a natural encoding of the complete tree is the vector $p = (p_1, p_2, \ldots, p_n)$, where p_i denotes the predecessor of node i. E.g. the predecessor vector of the spanning tree in Fig. 1 is p=(0,1,1,0,4,5,0,7,0).

Note that such a predecessor encoding has already been used in [Abuali et al., 1994] and discussed in [Palmer and Kershenbaum, 1994] and [Krishnamoorthy et al., 1999]. Especially in the latter two works, predecessor encoding has been pointed out to be problematic since an arbitrary vector p with $p_i \in \{0, 1, \ldots, n\}$ will not usually represent a feasible spanning tree but an unconnected graph containing cycles. This fact can simply be verified, since there are $(n+1)^n$ different predecessor vectors but only $(n+1)^{n-1}$ different spanning trees which are furthermore not all feasible solutions for the CMST problem. An EA using predecessor encoding in combination with traditional random initialization and variation operators such as one-point, two-point, or uniform crossover performs therefore poorly due to the huge amount of infeasible solutions that are generated.

4 Specialized Operators

We overcome the described disadvantage of the predecessor encoding by applying specialized initialization and variation methods which only generate solutions representing feasible spanning trees. In addition, all these methods include local heuristics.

4.1 Initialization

A solution p of the initial population is created by the following algorithm:

1. Let S be the set of already connected nodes to which other nodes may be connected and U the

set of yet unconnected nodes. Initially, S contains the root node 0 and U all client nodes.

2. While U is not empty:
   2.1. Select a node j from S randomly.
   2.2. Try to select a random node i from U which can be connected to node j without violating the capacity constraints. If no such node exists, remove j from S and go to step 2.1 as long as S is not empty.
   2.3. If S becomes empty, the algorithm is not able to generate a feasible solution due to priorly fixed links; restart the whole algorithm at step 1.
   2.4. Connect i to j by setting $p_i = j$ and moving i from U to S.
   2.5. Continue at step 2.

In order to implement step 2.2 efficiently, information about the capacity available for connecting further subnodes is stored for each node in S. Furthermore, the selection of node i in step 2.2 is guided by a local heuristic to favor low-cost links: A series of α uniform random choices of nodes in U is performed, and the node with the smallest link costs $c_{i,j}$ is actually selected if the capacity constraints allow it. Otherwise, this node i is discarded from further consideration as descendant of node j and the selection process is repeated with the remaining nodes in U.

Note that in all test problem instances we used, the algorithm was able to create a feasible solution without performing any restart due to priorly fixed links according to step 2.3. In more detail, it can be guaranteed that no restart will be necessary if

$$d_i \leq m_{0,i}\,, \quad i = 1, \ldots, n \tag{3}$$

holds, i.e. connecting all client nodes directly to the root is a feasible solution. Since this condition also holds in typical practical applications, step 2.3 can be said to not degrade performance.

4.2 Edge-Crossover

The major design goals for this binary crossover operator were computational efficiency, to provide strong locality, i.e. a new solution should primarily be built with links already contained in the two parents, and to create new feasible solutions with high probability.

Given two parental predecessor vectors p and q, a new solution r is derived in the following way:

1. Initialize r by setting all elements to the value -1 indicating that no links are selected up to now.

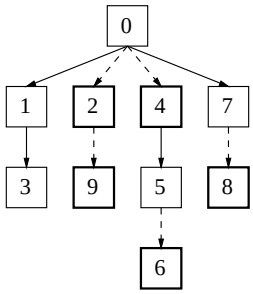

Figure 2: An exemplary offspring r=(0,0,1,0,4,5,0,7,2) created by recombining parents p=(0,1,1,0,4,5,0,7,0) and q=(0,0,1,7,4,1,0,0,2) with edge-crossover.

2. Adopt all links that are identical in both parents p and q, i.e. for all client nodes i, if $p_i = q_i$, also set $r_i \leftarrow p_i$.

3. Let U be the set of all client nodes having no predecessor yet, i.e. $U = \{i \,|\, r_i = -1\}$.

4. While U is not empty:
   4.1. Select a random node i from U.
   4.2. Choose $j \in \{p_i, q_i\}$ randomly.
   4.3. If nodes i and j are already connected somehow or the link (j, i) would violate any capacity constraints in r, go to step 4.4. Otherwise, add the link by setting $r_i \leftarrow j$ and removing i from U and continue at step 4.
   4.4. Try to adopt the predecessor of i from the other parental solution in the same way; if this is also not possible, move i from U to a queue Q storing nodes that cannot be connected via parental edges.
   4.5. Continue at step 4.

5. Finally, all nodes $i \in Q$ are processed as follows:
   5.1. Try to find a random node $j \in \{0, 1, \ldots, n\} \setminus \{i\}$ to which node i can be connected without introducing a cycle or violating any capacity constraint.
   5.2. If a feasible node j has been found, connect i to j by setting $r_i = j$ and continue at step 5 with the next node i.
   5.3. Otherwise, no feasible solution can be generated; make r a copy of p and terminate.

Figure 2 shows an exemplary offspring created from p=(0,1,1,0,4,5,0,7,0) and q=(0,0,1,7,4,1,0,0,2). This offspring r=(0,0,1,0,4,5,0,7,2) inherited edges (0,1), (1,3), (4,5), and (0,7) from both parents, edges (0,4),

(5,6), and (7,8) from parent p and edges (0,2) and (2,9) from parent q. Since after the completion of step 4 all nodes are connected to a feasible tree, it is not necessary to add entirely new edges (step 5) in this example.

A union-find data structure [Kingston, 1990] is used for efficiently checking whether two nodes are already connected via some link(s) or not (steps 4.3 and 5.1). This data structure is initialized with a set of $n+1$ independent elements representing all the nodes including the root. If a link is established in r, the corresponding elements of the data structure are united.

For an efficient check of the capacity constraints in steps 4.3 and 5.1, we furthermore store for each node i to which there has not yet been assigned a predecessor the total demand, i.e. the demand of the node itself and all its connected subnodes. The total demand of node i needs to be updated when any new node is connected to i or any of its subnodes.

If no feasible predecessor can be found for a client node (step 5.3), the crossover operator degenerates to a reproduction operator, but no infeasible solution is ever returned. This behavior might reduce the actual probability of producing new solutions by crossover in the EA significantly. However, in our empirical tests the amount of successful crossovers always remained relatively high.

As a local heuristic, low-cost edges are again favored when choosing the parental predecessor to try first (step 4.2): With a probability of 50%, the predecessor inducing the smaller link costs is tried first; otherwise, a uniform random decision is made. Furthermore, we also favor low-cost edges in step 5.1 by checking first the nodes being the source of the $n/8$ cheapest edges to node i.

4.3 Edge-Mutation

The aim of the mutation operator is mainly to introduce new links for avoiding premature convergence. The following algorithm tries to randomly modify a solution p in such a way that it remains feasible. In the worst case, i.e. when no feasible alternative can be found, the original solution remains unmodified.

1. Select a random client node $i \in \{1, \ldots, n\}$.
2. We think of removing the edge connecting node i with its predecessor p_i; this divides the tree into two unconnected components; determine the set S of all nodes that remain connected to the root by depth-first search (including the root node).
3. Remove p_i from S.
4. While S is not empty:
   - 4.1 Select a node j from S randomly.
   - 4.2 If the inclusion of edge (j, i) would violate any capacity constraint, remove j from S and continue at step 4.
   - 4.3 Actually include the edge (j, i) by setting $p_i \leftarrow j$; terminate with p as solution.
5. No alternative predecessor could be found for node i; return original solution p.

As a local heuristic, low-cost edges are again favored when selecting the new predecessor in step 4.1. This is done as in step 5.1 of the crossover operator by selecting first the nodes being the source of the $n/8$ cheapest edges to node i.

5 Empirical Results

This section contains a description of the remaining parts of the EA, the test data and the obtained empirical results.

We applied a steady-state replacement scheme, where in every generation not the whole population is renewed, but only one new individual is created by crossing over two selected parents and eventually performing mutation. Such a new solution replaces always the worst individual of the population with one important exception: If a solution is generated that is already contained in the population, this new solution is immediately discarded to avoid premature convergence [Raidl and Gottlieb, 1999]. The parents are selected via tournament selection.

Suitable strategy parameters for the EA were determined in preliminary test runs. For the results presented in this section, the following settings had been used:

- The population size was 500 individuals.
- The parameter α, which controls how strongly cheaper edges are preferred when generating initial solutions, was set to $n/2$.
- The crossover operator was always carried out.
- The mutation probability was set to 70%.
- The group size for the tournament selection was set to 5 individuals.
- As termination condition a convergence criterion was used: If no progress had been made (i.e. no better solution had been found) within the last 20,000 generations, a run was terminated.

Table 1: Results for the *tc* and *te* problem instances with $n = 40$ and $n = 80$.

Problem	$m_{i,j}$	C_{opt}		Best(C)	Avg(C)	$\sigma(C)$	t [s]	Best(gap)	Avg(gap)	ART: gap	EW: gap
tc40-1	3	742	opt	742	742.0	0.00	43.1	0.00	0.00	0.00	4.31
tc40-2	3	717	lb	717	718.1	1.52	47.8	0.00	0.15	0.98	4.46
tc40-3	3	716	lb	716	716.2	0.42	54.4	0.00	0.03	0.42	1.68
tc40-4	3	775	lb	778	779.3	1.70	51.0	0.39	0.55	0.65	3.74
tc40-5	3	741	opt	741	741.2	0.63	39.9	0.00	0.03	0.00	2.56
tc40-1	5	586	opt	586	587.0	1.05	32.0	0.00	0.17	0.34	1.54
tc40-2	5	578	lb	579	579.2	0.63	35.7	0.17	0.21	0.87	1.73
tc40-3	5	577	opt	577	577.0	0.00	36.4	0.00	0.00	0.69	4.33
tc50-4	5	617	opt	617	617.1	0.32	30.3	0.00	0.02	0.00	4.54
tc40-5	5	600	lb	602	604.5	1.08	34.5	0.33	0.75	0.83	2.50
tc40-1	10	498	opt	498	498.0	0.00	21.2	0.00	0.00	0.00	3.61
tc40-2	10	490	opt	490	490.4	0.84	24.6	0.00	0.08	0.41	3.06
tc40-3	10	500	opt	500	501.8	2.90	25.8	0.00	0.36	0.00	3.40
tc40-4	10	512	opt	512	512.4	0.70	22.3	0.00	0.08	0.39	2.34
tc40-5	10	504	opt	504	504.0	0.00	22.7	0.00	0.00	0.00	7.14
Average gaps								0.06	0.16	0.37	3.40
te40-1	3	1190	lb	1190	1191.1	0.88	91.4	0.00	0.09	0.08	2.10
te40-2	3	1103	lb	1103	1109.7	6.80	111.6	0.00	0.61	1.00	2.81
te40-3	3	1115	opt	1115	1115.0	0.00	69.6	0.00	0.00	0.36	2.78
te40-4	3	1132	lb	1134	1134.8	1.62	87.3	0.18	0.25	0.62	1.86
te40-5	3	1104	lb	1104	1108.1	1.97	130.2	0.00	0.37	1.09	3.89
te40-1	5	830	lb	830	833.9	2.18	87.8	0.00	0.47	2.05	3.25
te40-2	5	792	lb	792	797.4	8.11	90.2	0.00	0.68	1.01	5.39
te40-3	5	797	lb	801	806.0	3.77	113.1	0.50	1.13	2.76	2.89
te40-4	5	814	lb	814	822.0	4.81	75.6	0.00	0.98	2.21	4.91
te40-5	5	784	lb	784	784.4	1.26	96.6	0.00	0.05	1.15	4.08
te40-1	10	596	lb	596	599.4	4.99	50.6	0.00	0.57	5.70	10.40
te40-2	10	573	lb	581	581.0	0.00	40.7	1.40	1.40	0.70	10.30
te40-3	10	568	lb	568	569.1	1.66	51.6	0.00	0.19	1.94	4.93
te40-4	10	596	lb	596	597.6	0.84	37.8	0.00	0.27	1.34	7.05
te40-5	10	572	opt	572	575.2	1.93	51.3	0.00	0.56	1.22	4.37
Average gaps								0.14	0.51	1.55	4.77
Average over all instances with $n = 40$								0.10	0.34	0.96	4.09

Problem	$m_{i,j}$	C_{opt}		Best(C)	Avg(C)	$\sigma(C)$	t [s]	Best(gap)	Avg(gap)	ART: gap	EW: gap
tc80-1	5	1094	lb	1112	1124.9	8.97	262.1	1.64	2.82	3.75	8.04
tc80-2	5	1090	lb	1106	1111.3	2.54	250.9	1.47	1.95	3.76	7.34
tc80-3	5	1067	lb	1078	1089.8	5.05	268.0	1.03	2.14	2.72	6.00
tc80-4	5	1070	lb	1093	1100.4	5.80	297.1	2.15	2.84	4.39	7.57
tc80-5	5	1268	lb	1296	1306.8	6.68	293.3	2.21	3.06	2.52	5.52
tc80-1	10	878	lb	896	897.6	0.84	83.1	2.05	2.23	2.28	4.78
tc80-2	10	875	lb	889	891.1	2.02	92.0	1.60	1.84	1.83	4.80
tc80-3	10	869	lb	882	885.2	2.53	88.2	1.50	1.86	2.42	5.41
tc80-4	10	863	lb	876	876.0	0.00	93.0	1.51	1.51	3.36	6.03
tc80-5	10	998	lb	1029	1031.7	2.54	107.7	3.11	3.38	4.01	7.11
tc80-1	20	834	opt	838	841.2	1.69	58.9	0.48	0.86	0.48	2.64
tc80-2	20	820	opt	824	825.8	0.63	61.2	0.49	0.71	0.73	1.95
tc80-3	20	828	opt	828	831.2	2.53	69.1	0.00	0.39	1.45	3.38
tc80-4	20	820	opt	824	825.6	2.07	62.9	0.49	0.68	0.98	5.61
tc80-5	20	916	opt	928	936.6	7.18	69.8	1.31	2.25	2.73	6.00
Average gaps								1.40	1.90	2.49	5.48
te80-1	5	2531	lb	2571	2599.8	13.36	1100.7	1.58	2.72	1.34	2.88
te80-2	5	2522	lb	2596	2602.6	5.13	1489.6	2.93	3.20	1.94	4.40
te80-3	5	2593	lb	2647	2683.4	23.30	1609.6	2.08	3.49	2.93	5.01
te80-4	5	2539	lb	2577	2594.7	12.76	1491.0	1.50	2.19	1.89	3.35
te80-5	5	2458	lb	2487	2494.7	4.40	958.2	1.18	1.49	0.90	5.49
te80-1	10	1631	lb	1722	1754.3	18.63	265.0	5.58	7.56	3.92	7.05
te80-2	10	1602	lb	1659	1676.7	13.75	306.9	3.56	4.66	5.24	9.11
te80-3	10	1660	lb	1727	1742.6	18.57	361.0	4.04	4.98	4.58	10.12
te80-4	10	1614	lb	1652	1686.7	25.43	332.3	2.35	4.50	3.41	4.40
te80-5	10	1586	lb	1618	1655.1	23.41	347.5	2.02	4.36	3.91	7.94
te80-1	20	1256	lb	1291	1311.0	22.79	131.9	2.79	4.38	4.38	5.89
te80-2	20	1201	lb	1250	1260.8	5.18	159.9	4.08	4.98	4.41	7.33
te80-3	20	1257	lb	1280	1292.0	8.08	130.7	1.83	2.78	3.74	6.60
te80-4	20	1247	lb	1280	1290.8	9.95	152.6	2.65	3.51	6.26	7.70
te80-5	20	1231	lb	1248	1255.2	6.25	147.2	1.38	1.97	2.19	8.37
Average gaps								2.63	3.78	3.40	6.38
Average gaps over all instances with $n = 80$								2.02	2.84	2.95	5.93

We tested the described EA on a standard test data set, which we obtained from the OR-Library[1] of J. E. Beasley. This set has already been used in several other works, such as [Amberg et al., 1996, Ahuja et al., 1998, Rolland et al., 1998]. There are two categories of problem instances named *tc* and *te*. In all problem instances nodes have unit demands ($d_i = 1$) and links have identical capacities $m_{i,j}$. In both categories, there are five cost matrices for instances with $n = 40$ and $n = 80$ client nodes. Together with three different values for the link capacities this yields 30 ($= 5 \cdot 2 \cdot 3$) problem instances for each of the two categories *tc* and *te*.

The link costs $c_{i,j}$ are in both problem categories distances in the 2-dimensional Euclidean space, i.e. the costs satisfy the triangle inequality. In the *tc*-problems the server node was set near the center of all client nodes, whereas in the *te*-problems the server was placed somewhere near the border of the convex hull of the client node set. Because of this difference the *te*-problems have larger optimal objective values and are usually harder to solve to optimality.

Table 1 shows results obtained for all *tc* and *te* problem instances. 10 independent EA-runs were performed per problem instance. Column C_{opt} shows known optimum objective values or lower bounds as indicated (taken from [Rolland et al., 1998]). The third group of columns contains the best objective values out of 10 runs per instance, the average values, the standard deviations, and average CPU times in seconds on a 300MHz Intel Pentium-II PC. Note that these CPU times include the 20,000 generations needed for detecting convergence; thus the actual times for finding the best solutions are smaller.

In the last group of columns the results are given as percentage gaps to the values C_{opt} according to the formula $gap = (C/C_{\text{opt}} - 1) \cdot 100\%$. For comparison purposes, we also provide gaps obtained by the *Esau-Williams* (EW) heuristic and the *adaptive reasoning technique* (ART) as given in [Rolland et al., 1998].

The results for the problem instances of size $n = 40$ can be summarized as follows. In both problem categories, *tc* and *te*, the best as well as the average objective values achieved with the EA are significantly smaller than those of ART and EW. The average gap over all 30 problem instances is 0.34% in comparison to 0.96% of ART and 3.40% of EW. C_{opt} was reached at least once for 24 problem instances or in 80% of the cases. The EA's computational effort is about a factor of 2 higher for the *te*-problems than for category *tc*, but decreases in both cases as the link capacity $c_{i,j}$ increases. The higher CPU-times for the *te* instances emerge from the placement of the server nodes that makes these problems harder.

Considering the problem instances of size $n = 80$, the EA finds better solutions than EW and ART for category *tc*. For the *te*-problems the average gaps are slightly worse than those of ART, especially for $c_{i,j} = 10$. The average gap over all problem instances of size $n = 80$ is nevertheless smaller than that of ART. The computational effort for the *te*-problems is again higher, the factor varies – depending on the capacities – between about 2.5 and 5.

6 Conclusions and Further Work

In this article, we described an EA which can compete with one of the best heuristics for the CMST problem, the adaptive reasoning technique. Slightly better results have been achieved by the EA on a set of standard test problems. A careful design of the encoding and the specialized initialization and variation operators led to an EA that effectively searches in the space of feasible solutions only. In contrast to other encodings such as the Prüfer number encoding, this approach provides strong locality – an important basis for an efficient EA. Furthermore, the inclusion of local heuristics which promote the selection of cheap links turned out to be crucial.

Future work should include more tests using also other categories of problem instances, e.g. with nodes having unequal demands and edges having unequal capacities, to characterize the performance of the new approach in more detail.

Acknowledgments

This work is supported by the Austrian Science Fund (FWF) under the grant P13602-INF.

References

F. N. Abuali, D. A. Schoenefeld, and R. L. Wainwright. Designing telecommunications networks using genetic algorithms and probabilistic minimum spanning trees. In E. Deaton, D. Oppenheim, J. Urban, and H. Berghel, editors, *Proceedings of the 1994 ACM Symposium on Applied Computing*, pages 242–246. ACM Press, 1994.

R. K. Ahuja, J. B. Orlin, and D. Sharma. New neighbourhood search structures for the capacitated minimum spanning tree problem. Technical report, In-

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

dustrial and System Engineering Department, University of Florida, 1998.

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

S. Brave and A. S. Wu, editors. *Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference*, Orlando, FL, 1999.

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

I.-C. Chang and P. H. Vance. A branch-and-price algorithm for the capacitated minimum spanning tree problem. Technical report, Goizueta Business School, Atlanta, FL, 1998.

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

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.

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

L. Hall. Experience with a cutting plane algorithm for the capacitated spanning tree problem. *ORSA Journal on Computing*, 8(3):219–234, 1996.

J. R. Kim and M. Gen. Genetic algorithm for solving bicriteria network topology design problem. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, A. Zalzala, and W. Porto, editors, *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, pages 2272–2279. IEEE Press, 1999.

J. H. Kingston. *Algorithms and Data Structures – Design, Correctness, Analysis*. Addison-Wesley, Singapore, 1990.

J. Knowles and D. Corne. A new evolutionary approach to the degree constrained minimum spanning tree problem. *IEEE Transactions on Evolutionary Computation*, to appear, 2000.

M. Krishnamoorthy, A. T. Ernst, and Y. M. Sharaiha. Comparison of algorithms for the degree constrained minimum spanning tree. Technical report, CSIRO Mathematical and Information Sciences, Clayton, Australia, submitted to Journal of Heuristics, 1999.

Y. Li and Y. Bouchebaba. A new genetic algorithm for the optimal communication spanning tree problem. In C. Fonlupt, J.-K. Hao, E. Lutton, E. Ronald, and M. Schoenauer, editors, *Proceedings of Artificial Evolution: Fifth European Conference*, to appear as LNCS, Berlin, 1999. Springer.

C. C. Palmer and A. Kershenbaum. Representing trees in genetic algorithms. In D. Schaffer, H.-P. Schwefel, and D. B. Fogel, editors, *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 379–384. IEEE Press, 1994.

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

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.

P. Piggott and F. Suraweera. Encoding graphs for genetic algorithms: An investigation using the minimum spanning tree problem. In X. Yao, editor, *Progress in Evolutionary Computation*, LNAI 956, pages 305–314. Springer, Berlin, 1995.

R. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36: 1389–1401, 1957.

G. R. Raidl. An efficient evolutionary algorithm for the degree-constrained minimum spanning tree problem. In *Proceedings of the 2000 IEEE Congress on Evolutionary Computation*, to appear, 2000.

G. R. Raidl and J. Gottlieb. On the importance of phenotypic duplicate elimination in decoder-based evolutionary algorithms. In Brave and Wu [1999], pages 204–211.

G. R. Raidl and B. A. Julstrom. A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem. In J. Carroll, E. Damiani, H. Haddad, and D. Oppenheim, editors, *Proceedings of the 2000 ACM Symposium on Applied Computing*, pages 440–445. ACM Press, 2000.

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.

F. Rothlauf and D. Goldberg. Tree network design with genetic algorithms – An investigation in the locality of the Pruefernumber encoding. In Brave and Wu [1999], pages 238–243.

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

G. Zhou and M. Gen. A note on genetic algorithms for degree-constrained spanning tree problems. *Networks*, 30:91–95, 1997.