

A hybrid ACO algorithm for the Capacitated Minimum Spanning Tree Problem

Marc Reimann Marco Laumanns

Institute for Operations Research,
Swiss Federal Institute of Technology Zurich,
Clausiusstrasse 47, CH-8092 Zurich, Switzerland
email: {Marc.Reimann, Marco.Laumanns}@ifor.math.ethz.ch

Abstract

The problem of finding a Capacitated Minimum Spanning Tree asks for connecting a set of client nodes to a root node through a minimum cost tree network, subject to capacity constraints on all links. This paper reports on our design, implementation and performance evaluation of a hybrid Ant Colony Optimization algorithm for finding Capacitated Minimum Spanning Trees. Our Ant Colony Optimization algorithm is based on two important problem characteristics, namely the close relationship of the Capacitated Minimum Spanning Tree Problem with both the Capacitated Vehicle Routing Problem and the Minimum Spanning Tree Problem and hybridizes the Savings based Ant System with the algorithm of Prim, which is used to solve the subproblems of finding minimal spanning trees exactly. To assess the performance of our implementation we perform a computational study on a set of well known benchmark instances. For these instances our results show both the effectiveness and the efficiency of our algorithm when compared to several other state-of-the-art techniques.

1 Introduction

The problem of finding a Capacitated Minimum Spanning Tree (CMST) is one of the key problems in the design of telecommunication networks (see e.g. [1]). It asks for connecting a set of client nodes to a root node through a minimum cost tree network, subject to capacity constraints on all links. Thus, the solution will be a set of subtrees connecting clusters of client nodes to the root node. This problem is known to be NP-complete in the strong sense (see [2]). Thus, the development of heuristic and metaheuristic search algorithms for the CMST has been the focus of academic researchers.

Surprisingly, the CMST problem has not yet been tackled by Ant Colony Optimization (ACO), a metaheuristic inspired by the observation that real ants can find shortest paths and developed by Dorigo et al. [3, 4]. Recently, ACO has been successfully applied mainly to different routing problems (see e.g. [5]–[11]). Convergence proofs for generalized versions of ACO algorithms can be found in [12]–[14].

This paper reports on the design, implementation and performance evaluation of a hybrid ACO algorithm for the CMST. Our ACO algorithm is based on two important problem characteristics. First, the CMST problem is basically a relaxation of the Capacitated VRP (CVRP). In both problems nodes need to be clustered to comply with capacity requirements. However, in the latter problem a feasible solution for each cluster is not a subtree but a hamiltonian (travelling salesman) tour for a vehicle. Thus, by solving the CVRP we obtain an upper bound for the CMST solution. Second, given the clustering of client nodes in the CMST, the remaining subproblems consist in finding a Minimum Spanning Tree (MST) for each cluster, which is exactly solvable in polynomial time e.g. by the well known algorithms of Kruskal [15] or Prim [16]. We combine these

two characteristics of the CMST in our ACO algorithm in the following way. First, we construct solutions using the Savings based Ant System originally developed for the CVRP [9, 10]. Given the CVRP solution, we then compute an MST for each cluster by applying an implementation of Prim’s algorithm¹. Thus, the clusters are solved to optimality and the ants just need to find the optimal clustering. To assess the performance of our implementation we perform a computational study on a set of well known benchmark instances. For these instances our results show both the effectiveness and the efficiency of our algorithm when compared to several other state-of-the-art techniques.

The remainder of this paper is organized as follows. Section 2 describes the CMST and discusses related work, Section 3 presents our implementation of ACO for the CMST, Section 4 reports our computational experiments and analysis and Section 5 closes the paper with an outlook on future research issues.

2 Problem Description and Related Works

The CMST problem can be defined as follows. Given an undirected, fully connected graph $G = (V, E)$, where V is a set of n client nodes and a unique root node 0 and E is the set of possible edges. With each client node $i \in V \setminus \{0\}$ is associated a demand d_i , while with each edge $(ij) \in E$ a nonnegative cost c_{ij} is associated. Note, that in this paper we will deal with symmetric problems only, where $c_{ij} = c_{ji}$. Further, each edge (link) has a maximum capacity K . The problem consists in finding a minimum spanning tree (i.e., a tree with the minimum sum of edge costs), where for each subtree originating in the root node the total client node demand does not exceed the link capacity K .

This problem has been extensively studied in the past. In [17] several exact, approximative and heuristic approaches are reviewed and new metaheuristic algorithms based on Tabu Search and Simulated Annealing are proposed. Another Tabu Search algorithm for the CMST was proposed in [18].

Recently, two algorithms based on adaptive learning have been proposed. In [19] a memory adaptive reasoning technique inspired by Tabu Search was proposed. In this algorithm, a simple deterministic algorithm is used to repeatedly construct solutions to the CMST. This deterministic algorithm is applied to an augmented graph, where edges are dynamically excluded from consideration based on the concept of tabu tenure. The results obtained with this approach indicate its competitiveness with the other techniques at a very low computational expense.

An evolutionary algorithm based on a predecessor coding was presented in [20]. Starting from the leaf nodes in a feasible CMST solution, i.e. the nodes with degree one, by tracking the paths to the root node, the direct predecessor of each client node can be identified. Using this representation, edge based crossover and mutation operators are used. The empirical results presented in [20] indicate that the algorithm finds high quality, robust solutions, however at a quite high computational cost.

3 Ant Colony Optimization for the CMST

The approach proposed in this paper is based on the ACO metaheuristic originally developed by Dorigo et al. [3]. The underlying metaphor concerns the trail laying/trail following behavior of real ant colonies. Collective behavior in the exploitation of food sources results from the reinforcement of promising paths from the nest to a nearby food source.

The algorithm presented in this paper is in fact a modified version of an Ant System. The modification relates to the pheromone update and was first described in [21]. A high level description of the ACO algorithm for the CMST is given in Table 1.

¹As will be pointed out below we work with fully connected graphs. Thus, we use Prim’s rather than Kruskal’s algorithm as the former is more efficient for dense graphs.

```

procedure ACO for the CMST{
    Read the input data;
    Initialize parameters and pheromone matrix;
    repeat {
        for each ant {
            Construct a CVRP solution using the Savings based Ant System; (see 3.1)
            Improve the CVRP clustering by applying the Swap Local Search; (see 3.2)
            for each cluster {
                Compute MST using Prim's algorithm; (see 3.2)
            }
        }
        Update the best found solution (if applicable);
        Update the pheromone matrix; (see 3.3)
    } until a pre-specified stopping criterion is met;
}

```

Table 1: The ACO algorithm for the CMST

Let us now describe the basic structure of the algorithm in terms of solution construction and local search as proposed. Then we describe the pheromone initialization and updating rule.

3.1 Solution construction

As pointed out in the introduction, the CMST problem is closely related to the Capacitated VRP (CVRP). In both problems the client nodes have to be clustered due to capacity restrictions of the links (vehicles) in case of the CMST (CVRP). The only difference is that for each cluster the CMST problem asks for finding a Minimum Spanning Tree, while the CVRP requires each cluster to be solved as a TSP. Thus, while the clustering part of the two problems is identical, the subproblems are substantially more difficult to solve in the case of the CVRP and a solution of the CVRP will be an upper bound for the CMST.

Based on this observation, we employ a solution construction originally developed for the CVRP [9, 10] and based on the well known Savings algorithm [22]. At initialization all nodes are served directly from the root node. Thus, each node forms a separate cluster and the cost of the solution is just $c(S) = \sum_{i \in n} (c_{0i} + c_{i0})$, where c_{0i} denotes the cost of the link connecting the root node 0 with node i . Note that in this general formulation costs do not have to be symmetric.

Further, cost savings associated with merging two different clusters can be computed as

$$s_{ij} = c_{i0} + c_{0j} - c_{ij},$$

where s_{ij} denotes the cost saving associated with combining nodes i and j and merging the clusters nodes i and j belong to. Note that two nodes can only be combined if they are the end-nodes of two different clusters, i.e. if node i is visited last and node j is visited first in the sequence of the respective clusters. The initialization of the algorithm ends with sorting all possible savings values in decreasing order.

As in ACO algorithms the main working principle is to utilize a memory, let us now briefly describe how this memory is implemented. We use a square matrix τ with n columns and rows. Each entry τ_{ij} corresponds to the pheromone intensity on edge (ij) as an indicator of the usefulness of the combination of two nodes i and j in previous iterations, based on the evaluation of the solutions found.

In the iterative, constructive phase of the algorithm combinations of customers are chosen in the following probabilistic way, using both the savings and the pheromone information. Let a be the number of alternatives to consider in a given decision. Note, that a is a user defined parameter, which we will refer to as *neighborhood* throughout the paper. Furthermore, let Ω_a denote the set of edges (ij) associated with the a largest savings values corresponding to feasible cluster mergings

for a given decision. Note that this set changes from one decision to the next. In particular, an alternative becomes infeasible if it violates the capacity constraint K . The time complexity of computing the set Ω_a is $O(n)$. The decision rule can then be written as

$$\mathcal{P}_{ij} = \begin{cases} \frac{s_{ij}^\beta \tau_{ij}^\alpha}{\sum_{(hl) \in \Omega_a} s_{hl}^\beta \tau_{hl}^\alpha} & \text{if } (ij) \in \Omega_a \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

In equation (1) \mathcal{P}_{ij} gives the probability to combine nodes i and j into one cluster and α and β are two nonnegative integers that bias the relative influence of the pheromone trails and the savings values, respectively. Thus, in this decision rule the savings values correspond to the local heuristic rule about the usefulness of a node combination, while the pheromone concentration τ_{ij} corresponds to a global quality criterion. Higher savings and/or pheromone values imply a larger selection probability. This randomized decision process is repeated until no more feasible combinations are possible, i.e. the set Ω_a is empty. Using this approach, we work only with feasible solutions, i.e. for each cluster the capacity constraint is satisfied at all times during solution construction.

3.2 Solution improvement routines

The algorithm proposed in this paper hybridizes ACO with two different solution improvement procedures. The first one is a classic local search, namely the λ -interchange as proposed in [23] for the CVRP. Starting from a solution it tries to exchange at most λ subsequent customers on a route with a different sequence of up to λ customers. As soon as a possible improvement is detected, the associated λ -interchange is performed. This procedure is repeated until no more improvements are possible and the search ends in a local optimum. Note that λ can be zero for one of the two sequences. In this case, a sequence of customers is simply moved from its position to another position. Clearly as λ increases, the complexity of the local search grows quickly. Moreover, additional improvements in solution quality due to increasing λ are generally small. Thus, in the context of Tabu Search values of $\lambda \leq 2$ are typically considered, see e.g. [23].

The algorithm presented in this paper uses the λ -interchange in combination with both, the solution construction of the ants and another solution improvement technique. Thus, we will restrict its application to $\lambda = 1$, i.e. we only consider interchanges of single nodes. Further, we request that the two nodes to be interchanged belong to different clusters such that the application of this local search aims at improving the clustering of the solutions found by the ants. The time complexity of this 1-interchange is $O(n^2)$. In the context of ACO, this approach was already successfully applied to the CVRP in [9] and [10].

The second solution improvement mechanism used in this paper aims at the different clusters and exploits the characteristics of the CMST, whereafter for each cluster a Minimal Spanning Tree (MST) has to be found. The problem of finding an MST is quite easy and several efficient exact algorithms have been proposed for its resolution. In this paper we will use the well known algorithm of Prim [16]. Starting from an empty tree, i.e. no selected edges, it repeatedly includes the shortest edge connecting a node that is already part of the growing tree with a node that is not yet part of this tree. The algorithm ends once $n - 1$ edges have been included. The time complexity of an efficient implementation of this algorithm is $O(n^2)$, where n is the number of nodes.

In our algorithm we first apply the λ -interchange local search with $\lambda = 1$ to each ant's VRP solution before we obtain the optimal MST for each resulting cluster by using Prim's algorithm.

Note that to our best knowledge we are not aware of any work using an exact algorithm for the resolution of a subproblem within ACO. The only other paper we know of, that hybridizes ACO with exact algorithms is [21], where information from a problem relaxation is used as local heuristic to guide the solution construction.

3.3 Pheromone initialization and update

In the constructive phase of the ACO algorithm decisions are based on both heuristic information and the pheromone values as described above. At the end of each iteration, i.e. once all ants have gone through solution construction and local search, the pheromone update procedure is applied to these pheromone values. As pointed out above our update is a variant of the approach presented in [21]. It can be written as

$$\tau_{ij} := \rho\tau_{ij} + (1 - \rho)\Delta\tau_{ij}^* \quad \forall (ij) \in E \quad (2)$$

where $0 \leq \rho \leq 1$ is called the trail persistence and $\Delta\tau_{ij}^*$ is the amount of reinforcement. Let S^* be the best solution found up to the current iteration (regardless if it was found in the current iteration or earlier) and let $f(S^*)$ denote the objective value of this best found solution. Then $\Delta\tau_{ij}^*$ is defined as

$$\Delta\tau_{ij}^* := \begin{cases} f(S^*) & \text{if } (ij) \in S^* \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

In most standard ACO implementations (for minimization problems) this function is chosen as $f(S^*) = 1/c(S^*)$, where $c(S^*)$ is the cost of the solution. However, in this study we use $f(S^*) = 1$. The advantage of this approach is that pheromone values are unit-free and thus independent of monotonous transformations of the objective value.

The update strategy presented above is a pure elitist strategy, where for edges not belonging to S^* no reinforcement takes place, and the pheromone on these edges just evaporates at the rate $(1 - \rho)$ towards zero. On the other hand setting $\Delta\tau_{ij}^* = 1$ for links that are part of the best found solution implies that the pheromone on these links converges to 1.

Together with the fact that at the beginning of the run, the pheromone values are initialized to 1, i.e.

$$\tau_{ij} = 1 \quad \forall (ij) \in E, \quad (4)$$

the pheromone values now have a well defined domain, namely $\tau_{ij} \in [0, 1]$. Note that a more generalized version of this update is the so called Hypercube Framework proposed in [24]. Finally, the pheromone initialization strategy used in our approach leads to a broad search of the algorithm in the initial phase of our run and was originally proposed in the context of the Max-Min Ant System in [11].

4 Computational Experiments

The algorithm proposed above was coded in C and run on a Pentium with 1.5 GHz. The following parameters were chosen for the ACO algorithm: a population of n ants, $\alpha = \beta = 1$, $\rho = 0.975$ and $a = n/4$. Further, we used a static stopping condition by setting the maximum number of iterations to $maxIterations = 10n$. Note, that these parameters were chosen in accordance with previous experience from applying the Savings based Ant System to the CVRP (see [9]).

In this section we present results for a set of benchmark instances taken from the OR-Library² maintained by J.E. Beasley. We used 10 instances with 40 client nodes and a separate root node each. Client nodes are randomly scattered in a square grid of dimension 100-by-100. All client nodes have unit demand, i.e. $d_i = 1$ and link costs are symmetric, i.e. $c_{ij} = c_{ji}$, and the integer part of the Euclidean distance between nodes i and j . In five of these instances the root node is centered among the client nodes (denoted as *tc*), whereas in the other five instances the root node is located at the corner of the grid (denoted as *te*). For each instance three variants of link capacities are considered, namely $K = 3, 5$ and 10 units. Thus, we end up with 30 different problem scenarios.³

²<http://mscmga.ms.ic.ac.uk/info.html>

³Note, that results from a much more extensive numerical analysis can be found in a technical report [25]. This technical report as well as the ACO algorithm can be obtained from M. Reimann upon request.

To each problem scenario we applied our ACO algorithm 10 times and below we will report best, average and worst solution quality obtained, as well as average computation times. In Table 2 we compare our results with best results from several competing algorithms. These are the Tabu Search (denoted by TS) presented in [18], the Adaptive Reasoning Technique (ART) from [19], the Evolutionary Algorithm (EA) proposed in [20]. Further, we replicate results for the algorithms proposed in [17] (ADV) and the lower bounds (LB) proposed in [26] for all instances as presented in [19]. Note, that the LBs were obtained by Langrangean Relaxation. The names of the problem instances are shown in the first two columns of Table 2, i.e. the name of the first instance is *te1*.

type	#	K	LB	ACO					TS	ART		EA		ADV
				best	best %	avg. %	worst %	t[s]	%	%	t[s]	%	t[s]	%
te	1	3	1190	1190	0	0.08	0.08	2.4	0.17	0	11.5	0	91.4	n/a
te	2	3	1103	1103	0	0	0	0.0	1.27	0	11.2	0	111.6	n/a
te	3	3	1115	1115	0	0	0	0.4	0	0.18	11.5	0	69.6	n/a
te	4	3	1132	1134	0.18	0.18	0.18	2.3	1.06	0.18	11.2	0.18	87.3	n/a
te	5	3	1104	1104	0	0	0	4.6	1.00	1.00	11.3	0	130.2	n/a
tc	1	3	742	742	0	0	0	1.0	0.27	0	11.3	0	43.1	n/a
tc	2	3	717	717	0	0.17	0.56	1.2	1.53	0	11.1	0	47.8	n/a
tc	3	3	716	716	0	0.07	0.42	1.9	0.84	0	11.1	0.5	54.4	n/a
tc	4	3	775	775	0	0	0	0.4	2.32	0	11.1	0	51.0	n/a
tc	5	3	741	741	0	0	0	1.0	0	0	11.1	0	39.9	n/a
te	1	5	830	830	0	0.29	1.08	6.4	5.42	0.60	12.7	0	87.8	0
te	2	5	792	792	0	0.05	0.25	5.2	2.53	0.25	12.0	1.4	90.2	0
te	3	5	797	797	0	0	0	4.1	3.14	0	12.5	0	113.1	0
te	4	5	814	814	0	0	0	4.7	2.58	0.12	12.2	0	75.6	0
te	5	5	784	784	0	0	0	1.5	1.53	1.66	12.1	0	96.6	0
tc	1	5	586	586	0	0	0	0.6	0.68	0.34	11.8	0	32.0	0
tc	2	5	578	578	0	0	0	1.1	1.21	0.87	11.7	0	35.7	0
tc	3	5	577	577	0	0	0	0.6	0	0	12.0	0	36.4	0
tc	4	5	617	617	0	0	0	0.2	0.16	0	11.9	0.39	30.3	0
tc	5	5	600	600	0	0.07	0.33	2.3	0.33	0.83	12.0	0	34.5	0
te	1	10	596	596	0	0.64	0.84	3.6	3.02	2.01	14.4	0	50.6	0
te	2	10	573	573	0	0	0	4.3	3.14	0	12.4	0.17	40.7	0
te	3	10	568	568	0	0.04	0.18	3.0	4.05	0.70	12.3	0	51.6	0
te	4	10	596	596	0	0	0	1.3	2.01	0	12.5	0	37.8	0
te	5	10	572	572	0	0.03	0.35	3.7	0	0	12.4	0.33	51.3	0
tc	1	10	498	498	0	0	0	0.4	0.40	0	12.1	0	21.2	0
tc	2	10	490	490	0	0	0	0.1	0	0	12.1	0	24.6	0
tc	3	10	500	500	0	0	0	0.3	0	0	12.3	0	25.8	0
tc	4	10	512	512	0	0	0	0.7	0.20	0	12.0	0	22.3	0
tc	5	10	504	504	0	0	0	0.7	0	0	12.4	0	22.7	0
Avg.					0.01	0.05	0.14	2.0	1.30	0.29	11.9	0.10	56.9	0

Table 2: Comparison of results for *tc* and *te* instances with n=40 nodes

Let us briefly summarize the main findings from Table 2. First, we observe that our ACO algorithm shows excellent performance when compared to the lower bound. In fact, only for one instance we do not match the lower bound. It is worth mentioning at that point, that no other algorithm is able to hit the lower bound for this instance. Second, for 20 out of 30 instances we matched the LB in all runs. Thus, only for ten instances we observe runs that do not end with the optimal solution. Third, in all these cases the deviations are small with a worst case deviation of 1.08 % for one instance and overall performance of 0.05 % and 0.14 % deviation from the LB in the average and the worst case, respectively. Third, our computation times are on average 2 seconds, which also underpins the computational efficiency of our approach.

When looking at the competing approaches we observe that our algorithm is at par with the ADV results and slightly better than the EA while the ART and particularly the TS algorithm are clearly worse. Some issues have to be pointed out here, however. First, the ADV results are best solution values obtained over many runs with different algorithms, particularly Tabu Search and Simulated Annealing. Thus, the quality of a particular algorithm and its consumption of computation time can not be presented here. However, according to [17] the algorithms were

implemented in Fortran, run on a 486 processor with 66 MHz, the time limit was set to 600 seconds and the maximal time of the last improvements was reported to be close to this time limit.

Second, the EA results reported are best results over 10 runs and computation times are related to a Pentium 2 with 300 MHz. The average deviation presented for the EA in [20] is 0.34 % which is clearly above our worst case deviation. Thus, the ACO algorithm seems to outperform the EA both with respect to solution quality and computation times, even if we take the differences between the machines into account as proposed in [27]. Third, the results for the ART approach reported are best solution values over 100 runs for each instance and computation times relate to a Pentium 2 with 266 MHz. Again our algorithm clearly outperforms ART with respect to solution quality, while the computation times seem to be quite similar. Finally, the Tabu Search algorithm is clearly outperformed by all other algorithms.

Let us now look a little closer at problem characteristics and algorithm performance. First, we observe that the relative performance of our algorithm seems to slightly improve as capacity K increases. Clearly in these cases the subproblems become more important as there are fewer clusters with more nodes. Thus, solving these clusters exactly in the local search phase gives a great plus and helps the ants to concentrate on the clustering. On the other hand, for small capacities, more work is left to the ants to determine the ‘optimal’ clustering. Second, considering the location of the root node, we can support the observations from [17] and [19] whereafter the te instances are more difficult to solve than the tc instances. According to [28] this seems to relate to the fact that a non-centered root node effectively tightens the capacity constraints. However, note again that all the deviations from the lower bound produced by our ACO algorithm are very small such that these observations have to be confirmed on larger, more difficult instances.

5 Conclusions and outlook for further research

In this paper we have proposed an ACO algorithm for the Capacitated Minimum Spanning Tree problem. Its main characteristic is that it combines a solution construction developed for a more complicated problem, namely the Savings algorithm for the Capacitated VRP, with a Local Search that incorporates an exact algorithm for a subproblem, namely the Prim algorithm for finding Minimum Spanning Trees.

The computational study performed to assess the quality of the Ant Colony Optimization metaheuristic showed that our algorithm finds high quality solutions for all the benchmark instances and that it outperforms several other algorithms using comparatively little computational time. In fact 29 of the 30 instances were provably solved to optimality and for the last one the best known solution was matched.

Starting from the work presented in this paper, a number of possible paths for future research can be pointed out. For example, it should be interesting to replace the CVRP based solution construction with the Esau-Williams solution construction [29] which is custom made for the CMST problem. In fact it is an adaptation of the Savings algorithm, where savings between a node i and a node j are computed as the difference between the costs of the direct link between i and the root node and the link between i and j . Note, that in this case even a symmetric cost matrix implies asymmetric savings values between i and j , i.e. $s_{ij} \neq s_{ji}$. However, the disadvantage of this solution procedure is that the savings values cannot be computed once and for all in the initialization phase. Rather, (some of) these values change after each iteration, making the algorithm computationally more expensive. On the other hand, working directly with trees (rather than TSPs) in the solution construction may make the exact computation of spanning trees in the improvement phase obsolete. As this reduces computational effort, the overall effect is difficult to predict and thus worth to be studied.

These efficiency considerations become particularly relevant if the algorithm is to be applied to more realistic problem instances in terms of problem size and node demand. Particularly, large problems with non-unit demand will be an interesting area of research.

Note, that these issues are easily incorporated into the algorithm and from experience with the CVRP (for which demands are naturally different between nodes) it can be expected that the non-

unit demand case should not really degrade the performance of our ACO approach. Further, for very large scale instances it seems promising to apply our divide and conquer scheme as proposed in [10] for large scale VRPs.

Besides these issues some other interesting questions relate to the evaluation of other and possibly multiple objectives. For example, it might be interesting to try to find the ‘optimal’ root node, i.e., to simultaneously design the network and place the root node, or to adjust network connections in case of failure, i.e., to solve the dynamic problem when link capacities change or certain links fail completely. Finally, the multi-objective character of the problem is worth studying. Besides the direct costs of the network design, other objectives are the ‘optimal’ capacity of links, the maximum robustness of the system measured e.g. in terms of the maximum number of links from any given client node to the root node, or the maximum degree of the client nodes.

Acknowledgements

Comments of two anonymous referees which helped to improve the presentation of the paper significantly are gratefully acknowledged. We wish to thank Karl Doerner for his valuable support in implementing and testing the algorithm.

References

- [1] B. Gavish, *Topological design of telecommunication networks - local access design methods*, Annals of Operations Research **33** (1991), 17–71.
- [2] M.R. Garey and D.S. Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*, W.H. Freeman, New York (1979).
- [3] M. Dorigo and T. Stuetzle, *Ant Colony Optimization*, MIT Press/Bradford Books, Cambridge, MA (2004).
- [4] M. Dorigo, V. Maniezzo and A. Coloni, *The ant system: optimization by a colony of cooperating agents*, IEEE Transactions on Systems, Man, and Cybernetics-Part B **26(1)** (1996), 29–41.
- [5] B. Bullnheimer, R.F. Hartl and C. Strauss, *A new rank-based version of the ant system: a computational study*, Central European Journal of Operations Research **7 (1)** (1999), 25–38.
- [6] B. Bullnheimer, R.F. Hartl and C. Strauss, *Applying the ant system to the vehicle routing problem*, Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization (S. Voss et al., ed.), Kluwer, Boston, 1999.
- [7] M. Dorigo and L. M. Gambardella, *Ant colony system: a cooperative learning approach to the traveling salesman problem*, IEEE Transaction on Evolutionary Computation **1** (1997), 53–66.
- [8] L.M. Gambardella, E. Taillard and G. Agazzi, *Ant colonies for vehicle routing problems*, vol. New Ideas in Optimization, McGraw-Hill, 1999.
- [9] M. Reimann, M. Stummer and K.F. Doerner, *A savings based ant system for the vehicle routing problem*, Proceedings of GECCO2002, Morgan Kaufman Publishers, 2002, 1317–1325.
- [10] M. Reimann, K.F. Doerner and R.F. Hartl, *D-ants: Savings based ants divide and conquer the VRP*, Computers and Operations Research **31(4)** (2004), 563–591.
- [11] T. Stuetzle and H. Hoos, *MAX-MIN Ant System*, Future Generation Computing Systems **16(8)** (2000), 889–914.

- [12] W. J. Gutjahr, *A graph-based Ant System and its convergence*, Future Generation Computing Systems **16(8)** (2000), 873-888.
- [13] W. J. Gutjahr, *ACO algorithms with guaranteed convergence to the optimal solution*, Information Processing Letters **82** (2002), 145-153.
- [14] T. Stuetzle and M. Dorigo, *A short convergence proof for a class of ACO algorithms*, IEEE Transactions on Evolutionary Computation **6(4)** (2002), 358-365.
- [15] J.B. Kruskal, *On the shortest spanning subtree of a graph and the traveling salesman problem*, Proceedings of the American Mathematical Society **7** (1956), 48-50.
- [16] R.C. Prim, *Shortest connection networks and some generalizations*, Bell Systems Technology Journal **36** (1957), 1389-1401.
- [17] A. Amberg, W. Domschke and S. Voß, *Capacitated Minimum spanning Trees: Algorithms using intelligent search*, Combinatorial Optimization: Theory and Practice **1 (1)** (1996), 9-39.
- [18] Y.M. Sharaiha, M. Gendreau, G. Laporte and I.H. Osman, *A tabu search algorithm for the capacitated shortest spanning tree problem*, Networks **29** (1997), 161-171.
- [19] R. Patterson, H. Pirkul and E. Rolland, *A memory adaptive reasoning technique for solving the capacitated minimum spanning tree problem*, Journal of Heuristics **5** (1999), 159-180.
- [20] G.R. Raidl and C. Drexel, *A predecessor coding in an evolutionary algorithm for the capacitated minimum spanning tree problem*, Proceedings of GECCO2000, Late-Breaking-Papers, Las Vegas, 2000, 309-316.
- [21] M. Reimann, *Combining an exact algorithm with an Ant System for Travelling Salesman Problems*, Working Paper, University of Vienna, 2003.
- [22] G. Clarke and J.W. Wright, *Scheduling of vehicles from a central depot to a number of delivery points*, Operations Research **12** (1964), 568-581.
- [23] I.H. Osman, *Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem*, Annals of Operations Research **41** (1993), 421-451.
- [24] C. Blum and M. Dorigo, *The Hyper-Cube framework for ant colony optimization*, IEEE Transactions on Systems, Man and Cybernetics B **34(2)** (2004), 1161-1772.
- [25] M. Reimann and M. Laumanns, *Solving the Capacitated Minimum Spanning Tree Problem using Ant Colony Optimization*, IFOR - Technical report, ETH Zurich, 2004.
- [26] L. Gouveia, *A $2n$ constraint formulation for the capacitated minimal spanning tree problem*, Operations Research **43(1)** (1995), 130-141.
- [27] J.J. Dongarra, *Performance of Various Computers Using Standard Linear Equations Software, (Linpack Benchmark Report)*, Computer Science Technical Report CS-89-85, University of Tennessee, 2004.
- [28] L. Hall, *Experience with a cutting plane algorithm for the capacitated spanning tree problem*, INFORMS Journal on Computing **8(3)** (1996), 219-234.
- [29] L.R. Esau and K.C. Williams, *On teleprocessing system design. Part II - A method for approximating the optimal network*, IBM Systems Journal **5(3)** (1966), 142-147.

