



ELSEVIER

European Journal of Operational Research 124 (2000) 360–376

EUROPEAN  
JOURNAL  
OF OPERATIONAL  
RESEARCH

www.elsevier.com/locate/dsw

Theory and Methodology

# Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees

Anita Amberg<sup>a</sup>, Wolfgang Domschke<sup>a</sup>, Stefan Voß<sup>b,\*</sup>

<sup>a</sup> Technische Universität Darmstadt, Fachgebiet Operations Research, Hochschulstraße 1, D-64289 Darmstadt, Germany

<sup>b</sup> Technische Universität Braunschweig, Abteilung Allgemeine Betriebswirtschaftslehre, Wirtschaftsinformatik und Informationsmanagement, Abt-Jerusalem-Straße 7, D-38106 Braunschweig, Germany

Received 1 October 1996; accepted 1 March 1999

## Abstract

In the capacitated arc routing problem with multiple centers (M-CARP) the objective is to find routes starting from the given depots or centers such that each required arc is served, capacity (and usually additional) constraints are satisfied and total travel cost is minimized. In this paper we consider a heuristic transformation of the M-CARP into a multiple center capacitated minimum spanning tree problem with arc constraints, that we call arc-constrained CMST. An algorithm for determining initial feasible solutions as well as an improvement procedure for this problem are described and the “re-translation” of CMST solutions into the CARP context is explained. It is shown that the objective function value of the obtained CARP solution is easily derived from the respective value of the corresponding heuristic CMST solution. Furthermore, the possibility of including side constraints and the consideration of additional objective functions is discussed. Computations on real-world benchmark problems compare the results of the tabu search and simulated annealing metastrategies embedded in the improvement procedure. © 2000 Elsevier Science B.V. All rights reserved.

*Keywords:* Arc-routing; Capacitated minimum spanning tree problem; Tabu search

## 1. Introduction

In this paper we consider the capacitated arc routing problem (CARP) which belongs to an important class of vehicle routing problems with a great variety of real-world applications. In general,

the CARP can be characterized as follows: A *set of customers* has to be served by a *fleet of vehicles* operating from one or more *depots*. Each vehicle starts and ends its route at the depot it is assigned to. Furthermore, it has given *capacities* with respect to *time* (i.e. a maximal allowed time duration) and *quantity* (e.g., a maximal allowed capacity for satisfiable demand). Additional *side constraints* may exist, e.g., restricting the assignment of customers to vehicles. The objective is to find *routes* for the vehicles where each route sat-

\* Corresponding author. Fax: +49-531-391-8144.  
E-mail address: stefan.voss@tu-bs.de (S. Voß).

isfies the capacity constraints of the respective vehicle, each customer is served, and all side constraints are fulfilled such that the *total transportation cost is minimized*. Often, instead of optimizing transportation cost, the objective is to minimize transportation time or distance.

Customers, depots and streets may be represented by a graph  $G=(V, A)$  with node set  $V$  and arc set  $A$ . Arc routing applications refer to problems where the distribution or collection of goods is bound up with traversing a distance such as mail delivery, snow removal and winter gritting, garbage disposal, street sweeping and police patrols. Thus, the customers are modelled as arcs or edges, whereas in node routing problems the customers correspond with the nodes as, e.g., in the traveling salesman problem. The well-known Chinese postman problem (CPP) is the basic arc routing problem named after the Chinese scientist Mei-Ko Kwan (1962) who was the first to publish on this problem. It may be used as a *relaxation* for capacitated arc routing problems.

Though polynomial algorithms for the CPP in undirected or directed graphs are available (cf. Edmonds and Johnson (1973)) already the CPP in mixed graphs (containing directed arcs as well as undirected edges) with some nodes having odd degree is NP-hard as shown by Papadimitriou (1976). Introducing additional constraints even in undirected or directed graphs usually yields NP-hard problems such as the capacitated Chinese postman problem, where the capacity of the postman is restricted, or the rural postman problem (RPP) where the set of required arcs (i.e. those arcs which need serving) need not be connected and has to be linked using non-required arcs. For an excellent up-to-date review of arc routing problems including an extensive treatment of the CARP see Assad and Golden (1995).

We distinguish the 1-CARP where all vehicles are stationed in the same node (the *depot* or *center*) and have identical capacity constraints, and the M-CARP with vehicles starting from multiple depots and/or having different capacity requirements. The underlying graph of a CARP may be directed, undirected or mixed, and the subgraph of required arcs that need serving may be connected but need not be.

With respect to developing solution methods, it is important to note that capacitated arc routing problems consist of two interdependent subproblems: The assignment problem which forms subsets or clusters of required arcs served by the same vehicle and the sequencing or routing problem which determines the sequence of serving the arcs.

For the 1-CARP a few so-called “parallel” algorithms exist which simultaneously consider these subproblems: the path-scanning algorithm of Golden et al. (1983) and the construct and strike heuristic of Christofides (1973), both modified by Pearn (1989), the parallel-insert method of Chapeau et al. (1984), the augment-insert algorithm of Pearn (1991) and the augment and merge procedure of Golden and Wong (1981) which uses the savings criterion of Clarke and Wright (1964).

Sequential algorithms either choose the assignment before computing the routes inside the formed subsets (cluster-first-route-second, CFRS) or construct a giant route that is broken into small routes (route-first-cluster-second, RFCS). CFRS methods are more suitable for node routing problems; some CFRS algorithms for arc routing problems are described in Liebling (1970) and Bodin (1975). The RFCS algorithm of Stern and Dror (1979) designs a giant route as for the CPP and cuts it into small paths. The algorithms of Liebman and Male (1974) for the 1-CARP, as well as Geppert (1986) and Zhu (1989) for the M-CARP split the giant route into cycles and use savings criteria for the recombination. Furthermore, Benavent et al. (1990) apply the generalized assignment problem for their heuristic within the CARP context.

The purpose of this paper is to investigate a route-first-cluster-second algorithm for the M-CARP on undirected graphs where the clustering phase uses the following two concepts. First, the problem of finding the specific routes is modelled as a modified version of the capacitated minimum spanning tree problem (CMST) and a heuristic is applied to yield initial solutions. Second, some metastrategies, namely simulated annealing and tabu search, will be applied to improve the tree solutions eventually leading to better routes. Section 2 describes the basic RFCS concept of the algorithm including the treatment of additional

side constraints, the algorithm is able to deal with. Furthermore, in this section we briefly introduce the metastrategies that we use and present a mathematical formulation of the arc-constrained CMST, which is used in our algorithm. Results of applications to a real-world problem dealing with the routing of vehicles for winter gritting are presented in Section 3, followed by some conclusions.

## 2. The route-first-cluster-second approach

In this section we describe a route-first-cluster-second approach for the M-CARP in undirected graphs. The serving vehicles may differ with respect to the center or depot they are assigned to, with respect to demand capacity and a maximal allowed time duration or other serving capabilities. The vehicles have to cover the demand of the customers such that the total travel time is minimized. The customers are represented by edges where each edge has a given travel time for serving or just traversing it. The basic principle of the algorithm is to compute a giant route first (i.e. a tour through all edges of a given graph starting and ending at the same node while relaxing the capacity restrictions). Then this route or tour is divided into small cycles and a savings procedure is applied to recombine these cycles to routes satisfying the capacity constraints eventually using additional edges. The basic concept of this approach has been described by Zhu (1989). We adopt his idea while explicitly giving reference to the well-known capacitated minimum spanning tree problem which has not yet been done in his work. As an additional feature not considered in Zhu (1989) we apply different tabu search strategies to solve subproblems. More specifically, our approach is outlined by means of the following steps, which will be discussed subsequently:

- First a giant route corresponding to a minimal Euler graph is computed.
- The CARP is transformed into an arc-constrained capacitated minimum spanning tree problem. Given a spanning tree rooted at a node  $\omega$ , the components of the forest derived by eliminating  $\omega$  are referred to as subtrees off the root node  $\omega$ . The *capacitated minimum spanning tree*

*problem* (CMST) is to find a minimum cost tree spanning all nodes of a given graph such that all subtrees satisfy a prespecified capacity constraint, limiting the number of nodes in the respective subgraphs. Then the objective of the CMST is to find a minimal spanning tree rooted at  $\omega$  such that the number of nodes in any subtree off the root is at most a given (integer) capacity. (For references on the CMST see, e.g., Amberg et al. (1996), Gavish (1983), Gouveia (1995), Gouveia and Martins (1996), and Hall (1996). In the more general non-unit demand case of the CMST we are given non-negative node weights and the sum of the node weights in any subtree off the root is at most a given capacity).

In the so-called arc-constrained CMST we consider a second type of capacity constraint. (The mathematical formulation given in Section 2.3 corresponds to a directed CMST formulation thus referring to arcs instead of edges, although the example given below is undirected).

- A heuristic solution for the arc-constrained CMST is computed. This heuristic is a two-stage procedure which starts by finding an initial feasible solution and then tries to improve it by means of a local search procedure applying recent metastrategies as simulated annealing and tabu search.
- From this CMST solution a respective CARP solution is derived.
- A route optimization procedure changes the obtained CARP solution marginally and, eventually, reduces the total time used by additional included edges.
- As will be pointed out in Section 2.2, there may be different transformations of the CARP into the CMST. Thus, the transformation need not be unique and various transformations can be tried. Then the best of the obtained CARP solutions is chosen.

Though in general applied to an undirected graph  $G = (V, E)$ , problems in mixed graphs with a low share of directed arcs can be treated, too, where additional side constraints take care of the direction of the directed arcs. The subgraph of required edges need not be connected. Each edge

$(i, j)$  has two weights, namely a demand  $d_{ij}$  and a travel time  $t_{ij}$ . The time for traversing and at the same time serving edge  $(i, j)$  is assumed to be equal to  $t_{ij}$ , but the algorithm could be easily modified to deal with different values (i.e. taking into account additional time necessary to serve edge  $(i, j)$ ). Correspondingly, each of the  $M$  vehicles, say vehicle  $m$ , has demand capacity  $D_m$  and a maximal allowed time duration  $T_m$ . Vehicle  $m$  is stationed in depot  $DEP_m$ , where it starts and ends its route  $R_m$ . (Without loss of generality we may assume  $DEP_m = m$ , eventually making sufficient copies of depot nodes with more than one vehicle and re-numbering the nodes.) The objective is to minimize the total time of the additional edges that are added to form the routes. Two additional objective functions balancing the utilization of the vehicles and accounting for customer priority can be included.

The following subsections show the details of the algorithm. For illustration, an example is introduced and used throughout the whole section. Finally, the inclusion of side constraints and objective functions is considered and some remarks on the solution quality are given.

2.1. Computing a giant route

To obtain a giant route in the first step of the RFCS approach a minimal Euler graph is computed. A graph is called an *Euler graph* when a route exists that covers each edge *exactly once*. Otherwise, a graph may be augmented by additional edges to become an Euler graph. A *minimal Euler graph* is an Euler graph such that the sum of the edge weights of all augmented edges to derive

an Euler graph is minimal. To derive an Euler graph either existing edges may be duplicated or edges from a different set of (e.g., non-required) edges may be added. With respect to the CARP, an edge is called required if it needs to be served and non-required, otherwise.

To compute a minimal Euler graph the algorithm uses a modified version of the polynomial CPP matching algorithm of Edmonds and Johnson (1973) which allows the consideration of non-required edges (see e.g., Pütz, 1979). The given connected, undirected graph  $G = (V, E, t)$  with node set  $V$ , edge set  $E$  and edge length  $t_{ij}$  for each edge  $(i, j) \in E$  is augmented by the set of so-called Euler edges to an Euler graph. (If  $G$  is not connected, it is augmented to a graph where every edge has an even degree, possibly consisting of several components each being an Euler graph. For simplicity we will use the terms Euler graph and Euler edges throughout the paper in the latter case, too.) The total time of this optimal set of Euler edges is a lower bound for the total time of the extra edges that are needed in a feasible CARP solution. Note that in the algorithm of Zhu (1989) as well as in other RFCS algorithms for the CARP solutions are obtained that always contain the set of Euler edges. This is a possible reason for suboptimality in these approaches because there exist problem instances such that not all Euler edges need to be part of an optimal solution, which may be seen from an example below.

Fig. 1 shows the undirected graph of a CARP example with six nodes 1, 2, ..., 6. Node 1 is the depot of vehicle 1 with demand capacity and a maximal allowed time duration  $(D_1, T_1) = (14, 20)$ , and node 2 is the depot of vehicle 2 with demand

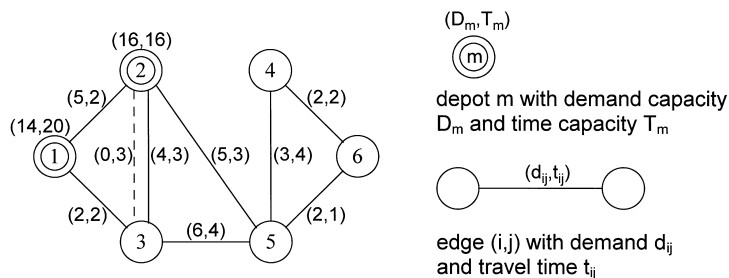


Fig. 1. Example.

capacity and a maximal allowed time duration  $(D_2, T_2) = (16, 16)$ . Each edge  $(i, j) \in A$  is given as solid line and has a label  $(d_{ij}, t_{ij})$  where  $d_{ij}$  is the demand of  $(i, j)$  and  $t_{ij}$  is the travel time for traversing  $(i, j)$ .  $G$  is connected but nodes 2 and 3 have odd degree. As can be easily seen, adding a copy of edge  $(2, 3)$  augments the graph to an Euler graph with minimal additional time of 3. In Fig. 1 this additional edge is drawn as a dotted line. Its weights are  $d_{23} = 0$  because serving is not required and  $t_{23} = 3$  because the time for traversing an edge is assumed to be independent from serving the edge. The obtained Euler graph has a total demand of 29 and a total travel time of 24. Thus, neither vehicle 1 nor vehicle 2 alone can perform the giant Euler tour without violating their capacity constraints.

As mentioned above Euler edges need not be part of an optimal solution. Assume  $(D_1, T_1) = (18, 21)$  and  $(D_2, T_2) = (11, 7)$  in the example of Fig. 1. Then, in an optimal CARP solution, edges  $(1, 2)$  and  $(1, 3)$  with cost 4 are used instead of edge  $(2, 3)$  with cost 3.

## 2.2. The transformation of the capacitated arc routing problem

In the next step the CARP is transformed into an arc-constrained CMST problem with an undirected or symmetric directed graph  $G' = (V', A', d', t')$ . First, the Euler graph is partitioned into a set of small sub-cycles of the giant route. Each of these cycles forms a node in  $V'$  and has demand and time weights corresponding to the sum of demand and time weights of the edges that are part of the respective cycle. Furthermore, each vehicle corresponds to a center node (or root node in the CMST terminology) with respective demand capacity and maximal allowed time duration. That is, in order to guarantee feasibility two different capacity constraints have to be observed with respect to the subtrees. Therefore, the arc-constrained CMST may be seen as a generalization of the unit demand CMST as well as the general demand CMST.

To summarize, we derive a multiple center arc-constrained CMST different from the original

CMST. Here, we have at most  $M$  root nodes (corresponding to depots) each with exactly as many subtrees as vehicles are stationed in the respective depot. Furthermore, there exists a capacity constraint on the sum of node and arc weights of the respective subtree (see Section 2.3).

Thus, in our example the graph for the arc-constrained CMST contains two root nodes: Node  $1'$  is the first center with capacity  $(D_{1'}, T_{1'}) = (14, 20)$  and node  $2'$  is the second center with capacity  $(D_{2'}, T_{2'}) = (16, 16)$ . The edge set of the Euler graph is split e.g., into three cycles forming three nodes  $3', 4'$  and  $5'$ : Node  $3'$  corresponds to the required edges  $(1, 2)$  and  $(1, 3)$  and the Euler edge  $(2, 3)$ . The capacity requirement of node  $3'$  is  $(d_{3'}, t_{3'}) = (7, 7) = (5, 2) + (2, 2) + (0, 3)$ . Node  $4'$  corresponds to the required edges  $(2, 3)$ ,  $(2, 5)$  and  $(3, 5)$  with a capacity requirement of  $(d_{4'}, t_{4'}) = (15, 10)$ . Node  $5'$  corresponds to  $(4, 5)$ ,  $(4, 6)$  and  $(5, 6)$  with  $(d_{5'}, t_{5'}) = (7, 7)$ .

Two other partitionings of the Euler graph could be derived: First the Euler edge and its original counterpart can be exchanged, both being parallel edges by definition. Second, these two edges between nodes 2 and 3 could form one cycle and edges  $(1, 2)$ ,  $(2, 5)$ ,  $(5, 3)$  and  $(3, 1)$  would form another. Provided that the partitionings are not unique, which is generally the case, in our algorithm the partitionings are chosen randomly.

The nodes of the CMST graph consist of cycles within the original augmented graph, i.e., it is called a *cycle node graph*. The arcs of the set  $A'$  represent cycles, too: An arc between two nodes  $i'$  and  $j'$  consists of a cycle of additional edges that connects the corresponding cycles of  $G$  and has minimal length (with respect to time). The demand weights of these arcs are zero because they consist of additional, i.e. non-required edges. If the cycles of two CMST nodes  $i'$  and  $j'$  have a common (CARP) node, within the original graph no connecting cycle is needed and  $t'_{ij} = 0$ . If no such common node exists, a connection between the two nearest neighbors of the two cycles is needed: for instance, with respect to the CMST center node  $1'$  and the non-center node  $5'$  the shortest connecting cycle in  $G$  is given by the edges  $(1, 2)$ ,  $(2, 5)$ ,  $(5, 2)$  and  $(2, 1)$  and has a total travel time of 10. The remaining arcs and their weights are deter-

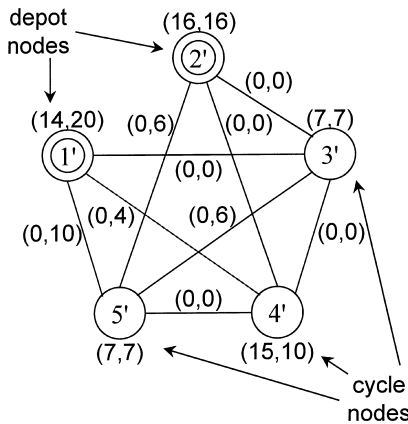


Fig. 2. The CMST graph  $G'$ .

mined in a similar way. Fig. 2 shows the resulting CMST graph. (Note that the graph is symmetric representing two arcs with the same weights for both possible directions.) The graph is complete apart from connections to (from) the center nodes. No root node is directly connected with another root because no route, and thus no subtree of the CMST, can include two centers.

In the CMST graph  $G'$  each node and each arc represents a cycle consisting of edges of the original graph  $G$ . Thus, each tree of the CMST can be “translated” into a route in the CARP. If the tree is feasible for the CMST, the route is feasible for the CARP.

Relaxing the capacity constraint of the CMST results in the problem of determining a minimum cost spanning forest consisting of exactly two components including the center nodes  $1'$  and  $2'$ , respectively, which can be obtained in polynomial time using a modified version of the algorithm of Kruskal (1956) or Prim-Dijkstra (cf. Prim, 1957; Dijkstra, 1959). Fig. 3 shows a (uncapacitated) minimum spanning tree (MST) of  $G'$ . It should be noted that the determination of a minimum cost spanning forest consisting of exactly two components including the center nodes  $1'$  and  $2'$ , respectively, does not satisfy the capacity requirements in our case. (A minimum spanning forest with zero cost consisting of exactly two components could be derived from the MST in Fig. 3, e.g., by eliminating edge  $(2', 3')$ ). That is, an

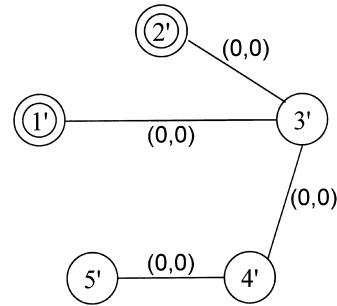


Fig. 3. A minimum spanning tree of  $G'$ .

explicit consideration of capacity constraints is inevitable.

Fig. 4(a) shows a feasible solution for the CMST. The subtree of center  $1'$  consists of arcs  $(1', 3')$  and  $(3', 5')$ ; arc  $(2', 4')$  forms the second subtree. The objective function value is 6 resulting from arc  $(3', 5')$  with weight  $t'_{3'5'} = 6$ . From the CMST solution a CARP solution can be derived where the total time of additional edges is 9 resulting from a cost of 3 for the Euler graph and 6 for the CMST. Fig. 4(b) shows the two subtrees of the CMST solution translated into the underlying CARP nodes and edges. Additional edges are drawn as dotted lines. For each subtree a feasible CARP route can be easily constructed.

In general, each feasible solution of the CMST results in a feasible CARP solution. The sum of the arc weights of the CMST solution plus the sum of the time weights of the Euler edges equals the sum of the time weights of the additional edges in the CARP solution.

### 2.3. The multiple center arc-constrained capacitated minimum spanning tree – a mathematical formulation

Before presenting a heuristic for the CMST in the next section, we present a mixed integer linear programming flow formulation for the multiple center CMST with non-equal node weights. Corresponding formulations for the one center CMST with equal node weights can be found in Gouveia (1995) and Gavish (1983). For related considerations with respect to a multiple center CMST see

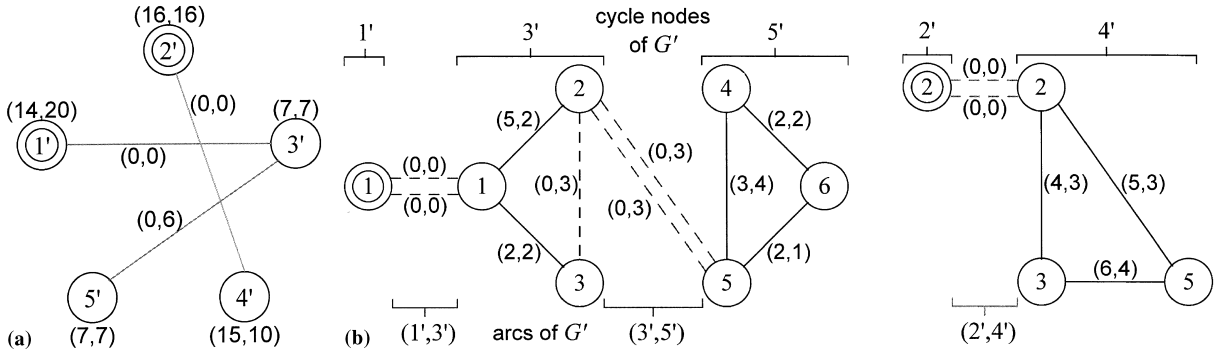


Fig. 4. (a) A feasible CMST for  $G'$ ; (b) The respective feasible CARP solution.

Gouveia and Lopes (1995), who define their problem in the context of telecommunications applications.

First, we consider only one capacity constraint concerning demand. Let  $G = (V, A, d, t)$  be the given graph with node set  $V = V_M \cup V_N$ , consisting of the set of center nodes  $V_M = \{1, \dots, M\}$  and the set of non-center nodes  $V_N = \{M + 1, \dots, n\}$ , arc set  $A$ , demand requirement  $d_h$  for each non-center node  $h \in V_N$  and travel time  $t_{ij}$  for each arc  $(i, j) \in A$ . (Note that in the arc-constrained CMST the demand requirement  $d_h$  is a node weight. If the CMST is derived from a CARP network the demand requirement of a node is computed as the sum of demand requirements of the edges included in the underlying CARP cycle.) Let the demand capacity of each center node  $m \in V_M$  be  $D_m$ . Define  $x_{ij} = 1$ , if arc  $(i, j)$  is included in the solution, and  $x_{ij} = 0$ , otherwise. Furthermore,  $y_{ij}$  denotes the flow on arc  $(i, j)$  for  $i = 1, \dots, n$  and  $j = M + 1, \dots, n$ . Ensure variables  $x_{ij}$  and  $y_{ij}$  with  $(i, j) \notin A$  to be equal to zero by assigning prohibitively large weights to the respective arcs. The following single flow formulation of the directed multiple center CMST gives a forest of  $M$  directed capacitated trees with the center nodes  $1, \dots, M$  being the roots:

$$\text{Minimize } Z = \sum_{i=1}^n \sum_{j=M+1}^n t_{ij} \cdot x_{ij} \quad (1)$$

subject to

$$\sum_{h=1}^n X_{hi} = 1, \quad i = M + 1, \dots, n, \quad (2)$$

$$\sum_{h=1}^n y_{hi} - \sum_{j=M+1}^n y_{ij} = d_i, \quad i = M + 1, \dots, n, \quad (3)$$

$$\begin{aligned} d_j \cdot x_{ij} &\leq y_{ij} \leq (D_{\max} - d_i) \cdot x_{ij}, \\ i &= M + 1, \dots, n, \text{ and} \\ j &= M + 1, \dots, n, \quad (i, j) \in A, \\ D_{\max} &= \max \{D_m | m = 1, \dots, M\}, \end{aligned} \quad (4a)$$

$$\begin{aligned} d_i \cdot x_{mi} &\leq y_{mi}, \quad m = 1, \dots, M \text{ and} \\ i &= M + 1, \dots, n, \end{aligned} \quad (4b)$$

$$\sum_{i=M+1}^n y_{mi} \leq D_m, \quad m = 1, \dots, M, \quad (4c)$$

$$\begin{aligned} x_{ij} &\in \{0, 1\}, y_{ij} \geq 0, \quad i = 1, \dots, n \text{ and} \\ j &= M + 1, \dots, n, \quad (i, j) \in A. \end{aligned} \quad (5)$$

Equalities (2) ensure that exactly one arc is reaching each non-center node. The flow conservation constraints (3) in common with the coupling constraints (4) attain the satisfaction of the capacity constraints and prevent cycles, thus guaranteeing a forest spanning all nodes. Contrary to formulations for the one center CMST, (1)–(5) allows for more than one central arc for each subtree. With (4c) the total flow of these central arcs is forced not to exceed the demand capacity of the respective center.

For the arc-constrained CMST we additionally introduce time weights  $t_h$  for each non-center node  $h$  as well as a maximal allowed time duration  $T_m$  for each center node  $m$  and claim that the total

time of each subtree  $m$ , i.e. the sum of time weights of the included nodes *and* arcs, must not exceed the given time capacity  $T_m$ . The above formulation is supplemented by time flow variables  $z_{ij}$  for all arcs  $(i, j)$  analogous to the flow variables  $y_{ij}$  and constraints (6)–(8) corresponding to (3)–(5):

$$\sum_{h=1}^n z_{hi} - \sum_{j=M+1}^n z_{ij} = \sum_{h=M+1}^n t_{hi} \cdot x_{hi} + t_i, \quad i = M + 1, \dots, n, \tag{6}$$

$$(t_{ij} + t_j) \cdot x_{ij} \leq z_{ij} \leq (T_{\max} - t_i) \cdot x_{ij}, \quad i = M + 1, \dots, n \text{ and } j = M + 1, \dots, n, \quad (i, j) \in A, \quad T_{\max} = \max\{T_m \mid m = 1, \dots, M\}, \tag{7a}$$

$$(t_{mi} + t_i) \cdot x_{mi} \leq z_{mi} T_m X_{mi}, \quad m = 1, \dots, M \text{ and } i = M + 1, \dots, n, \tag{7b}$$

$$\sum_{i=M+1}^n z_{mi} \leq T_m, \quad m = 1, \dots, M, \tag{7c}$$

$$z_{ij} \geq 0, \quad i = 1, \dots, n \text{ and } j = M + 1, \dots, n, \quad (i, j) \in A. \tag{8}$$

The multiple center arc-constrained CMST now is given by (1)–(8). The time flow  $z$  describes a flow of time capacity starting from the respective center node. The time flow variable  $z_{ij}$  contains the time capacity for arc  $(i, j)$ , node  $j$  and all arcs and nodes that are part of the remaining subtree beginning in node  $j$ . If in a subtree several arcs  $(i, j)$  part from the same node  $i$ , constraints (6) in common with (7) guarantee that the time capacity flow is split according to the capacity requirements of the respective succeeding subtrees. It should be noted that constraints (6) are similar to duration constraints in a routing model presented in Gavish and Graves (1979)

#### 2.4. A heuristic for the arc-constrained capacitated minimum spanning tree

The CMST with one center and one capacity constraint referring only to node weights has been

shown to be NP-hard by Papadimitriou (1978). Thus, the more general arc-constrained CMST is NP-hard, too. In such cases it is common practice to develop heuristic solution procedures. Furthermore, the argument for developing heuristics is strengthened due to the fact that using the CMST as a modeling approach within the CARP is heuristic by construction. A savings procedure based on the well-known algorithm of Esau and Williams (1966) produces an initial feasible solution. Then an exchange procedure using the metastrategies simulated annealing and tabu search is applied to improve the result.

*A savings procedure for the arc-constrained CMST:* For the arc-constrained CMST with multiple depots or centers a modification of the algorithm of Esau and Williams (1966) is applied. The modification concerns two points: first, the starting solution, and second, the way feasibility is determined with respect to the maximum allowed time duration and the demand capacity including both edge and node weights. The algorithm of Esau and Williams uses the so-called star tree as initialization, i.e., the spanning tree where each node is directly connected to the center node. In the arc-constrained CMST with multiple centers it is not known to which center a node is assigned to. Thus the savings procedure for the arc-constrained CMST introduces an artificial node  $X$  to the given CMST graph  $G' = (V', A')$ . Each non-center node  $i' \in V'$  is connected to  $X$  with an artificial arc  $(i', X)$ . The time weight of this arc is computed as the sum of time weights of all arcs leading from node  $i'$  to all center nodes. In our example the time weight of arc  $(5', X)$  is  $t_{5'X} = t_{5'1'} + t_{5'2'} = 10 + 6 = 16$ . We obtain  $|V'| - M$  non-central components  $C_{i'} = (V_{i'}, A_{i'}) = (\{i', X\}, \{(i', X)\})$ . Also each center defines a single component. Thus we have  $M$  central components  $C_m = (V_m, A_m) = (\{m\}, \phi)$ . The union of these components may be viewed as an artificial solution, which is of course infeasible for the CMST. It is changed stepwise by a savings algorithm: in each iteration two components are joined, i.e., one of the components is cut off the artificial node  $X$  and joined to the other component by the least cost edge (with respect to time). If both components were connected to  $X$  the artificial edge with maximum cost is deleted. The savings

are computed as the cost of the deleted arc minus the cost of the arc added to join the two components. In each iteration the joining which yields the highest savings is chosen.

The components that are formed have to fulfill the capacity requirements. If a non-center component (i.e. a component which does not contain any center node) is joined with a central component feasibility is easily checked. But if two non-center components are joined there is no center node and thus no capacity given. To avoid non-center components that are too big to be feasibly joined with a center each non-center component with a capacity requirement that exceeds half of the maximal free capacity of any center is immediately joined to a center component. After  $n$  joinings a feasible CMST solution is found.

*A node exchange improvement procedure for the arc-constrained CMST:* To improve the CMST solution we propose a *node exchange procedure* which transforms one feasible solution into a neighbor solution by changing the assignment of the nodes to subtrees. Such a transformation is called *move*. Subsequently a certain number of moves is executed thus trying to find a better – or even an optimal – solution. A metastrategy, like tabu search or simulated annealing as a *guiding process* decides which of the possible moves is chosen and forwards its decision to the *application process* which then executes the chosen move and, in addition, provides some information for the guiding process (depending on the requirements of the respective metastrategy) like the recomputed set of possible moves. What is meant to be a move depends on problem structure. In

our algorithm we consider so-called *shift moves*. A shift move chooses one node and shifts it from its actual component to another one. Fig. 5(a) shows two subtrees of a CMST. Now suppose node 4' is chosen and moved from subtree 1' (which includes center node 1') to subtree 2'. The resulting neighbor solution is given in Fig. 5(b). It can be seen that after changing the assignment of nodes the recomputation of minimum spanning trees inside the changed components becomes necessary.

Contrary to the above move definition we may describe an approach based on edge exchange. A move is defined as dropping one edge of the solution tree and adding another one. The gain of a move is computed as the respective cost difference. It is worth noting that the resulting subtrees need not be *minimum* spanning trees (defined on the corresponding subset of nodes), and although a reoptimization procedure is easy to implement, during the tabu search process the effect of a move is not given by the local cost changes implied by it. Thus, the search may favour a “wrong” direction. That is why we prefer defining a node exchange as a move (cf. Amberg et al., 1996).

Only *feasible moves* leading again to feasible solutions are allowed, but the cost of a new solution may exceed the cost of the previous one: moves leading to a cost increase are allowed in order to overcome local optima. Which of the available feasible moves should be chosen to transform the current solution? The answer to this question is not clear and various approaches may lead to good solutions. Zhu (1989) uses the metastrategy simulated annealing as a guiding process;

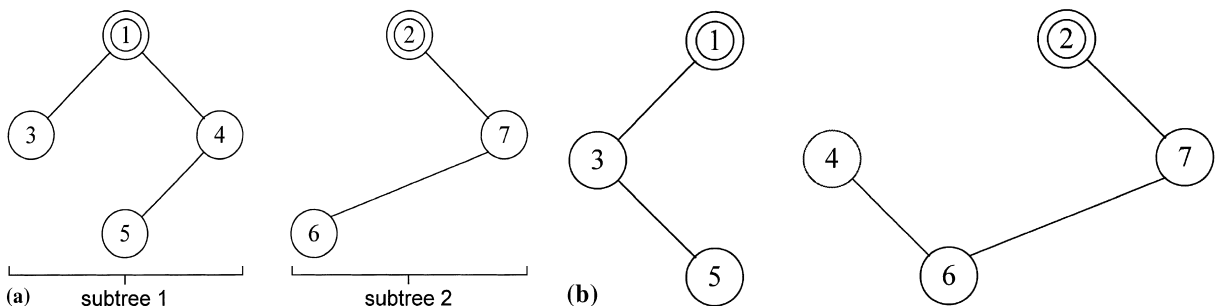


Fig. 5. (a) A CMST solution; (b) A neighbor solution.

our algorithm alternatively uses tabu search. The procedure stops after a given time limit.

### 2.5. Simulated annealing and tabu search

Simulated annealing randomly chooses one of the feasible moves and computes its change in cost (in our problem: travel time). If the change is a cost decrease the move is accepted. Otherwise, the new solution is accepted with a certain probability. The probability function usually is logarithmic and, in order to favour good solutions, decreases with larger cost increases. It decreases over time thus intensifying the search in the current area of the solution space as the computation time increases. Nevertheless, a parameter  $t$  called start temperature has to be specified to adapt the probability function to the actual problem. If the new solution is rejected the current solution remains unchanged in this step. The next iteration tries again to alter the same (current) solution. For more details see, e.g., Aarts and Korst (1989). Simulated annealing does not require any additional information.

Tabu search examines all feasible moves. The best move – which leads to the highest cost decrease or the lowest cost increase, respectively – is chosen and executed. Now suppose that a local optimum is reached. Without further instructions the algorithm could permanently alternate between this local optimum and its best neighbor. For that reason a so-called *tabu list* is created: to prevent an already explored solution from being examined again, all moves that (could) lead to such a solution are stored in the tabu list. Determining which moves have to be set tabu is derived from exploring the *running list*, which contains all performed moves in their sequence of execution. Both lists have to be updated after each iteration. It is worth noting that one need not store the attributes of the explored solutions, rather only the attributes of the moves, which usually requires less space.

In the literature there are several distinct ways of deriving the tabu list. They are referred to as static tabu search (STS), reverse elimination method (REM) and cancellation sequence method

(CSM) (cf. Glover, 1989, 1990). Some parameters have to be specified to adopt the methods to a specific problem and problem instances (especially problem size and scaling of cost). To be more specific and to make this paper self-contained some details of our tabu search implementations are given.

STS is a straightforward approach requiring a parameter  $s$  called *tabu list length*. STS disallows performing the complements of the  $s$  most recent moves, where the *complement* of a move is that move which cancels the effect of the considered one. That is, the tabu list (or tabu buffer) consists of a (complementary) copy of the  $s$  most recently performed moves. Older moves are disregarded. The tabu status derived from the  $s$  most recent moves forces the procedure to perform at least  $s$  moves before the explored solution may be reexplored. Obviously, this approach may disallow more moves than necessary to avoid returning to an already visited solution. This encourages keeping  $s$  as small as possible. However, if  $s$  is too small the procedure may return to a local optimum that was just left, i.e., cycling appears.

The REM and the CSM are based on cancellation sequences. In general, any connected part of the running list is a cancellation sequence (CS): the sequence of moves between two solutions. It may contain pairs of complementary moves. A residual CS (RCS) results from a CS by cancelling (i.e. eliminating) some of its moves. REM and CSM differ in the way they carry out these cancellations but they are the same in their fundamental principle: Any move that would cancel an RCS is set tabu. REM and CSM compute and maintain RCSs in different ways.

In each iteration REM recomputes all RCSs between the current and any previous solution. Each iteration starts at the current solution (with the tabu list and the RCS being empty) and then step by step traces in reverse direction through the running list. At each step an RCS between the current solution and the actually considered older solution is computed. Each time the RCS reduces to one move the respective complement is set tabu. It stays tabu only for the iteration immediately following. The parameter  $r$  is the maximal length of the running list and gives the maximum number

of iterations considered by the REM. If the running list is considered in its full length each move that should be tabu is found. Also each move that is set tabu would in fact lead to an already explored solution. Therefore, no better (or even optimal) solution can be “forgotten” because of being tabu without need.

The CSM embodies the concepts of the static method and the REM. Moves are set tabu in two ways: The static tabu process uses a tabu buffer  $c_3$  which is the length of the tabu buffer and at the same time the duration of the tabu status of the static tabu process. It corresponds to the tabu list length  $s$  of STS. The value  $c_3=1$  prevents the immediate return to the just visited solution. Higher values are reasonable, too, because the main CSM tabu process might not find every move that should be tabu. Usually  $c_3$  is smaller than  $s$  because of this additional way of setting tabu. The main tabu process like REM disallows any move that would cancel an RCS. Contrary to REM, the RCSs are stored and updated in each iteration: After the execution of an actual move CSM checks whether the complement of this move is part of the running list. (Parameter  $c_1$  is the maximal length of the running list and corresponds to parameter  $r$  of the REM.) If this is the case then two steps follow: First, a new RCS between these complementary moves is created and stored in an extra list already containing elder RCSs. Second, CSM checks whether the complement of this move is contained in any older RCS. If yes, it is cancelled from the respective RCS. Whenever this cancellation reduces the RCS to one move, say  $Y$ , to prevent the RCS from being completely cancelled, the complement of  $Y$  is defined tabu for the next  $c_2$  iterations.  $c_2$  is the duration of the tabu status of the main tabu process. Contrary to the REM this duration is usually more than one iteration. This is done because after setting the complement of  $Y$  tabu the CSM eliminates the respective so-called one-move RCS  $Y$  from further consideration.

In Amberg et al. (1996) we give a more detailed description of the tabu search methods with respect to the (unit weight) CMST. We restrict ourselves to consideration of these tabu search implementations, as they seem to perform quite

well on a number of well-known benchmark unit weight instances from the literature, as confirmed by Gouveia and Martins (1996) and Hall (1996). According to Gouveia and Martins (1996) they are even able to outperform other known heuristics of good quality such as second order heuristics (see e.g., Kershenbaum et al., 1980) and also the method of Sharaiha et al. (1997) which is a tabu search approach, too.

## 2.6. A route optimization procedure

After applying the improvement procedure the obtained CMST solution is transformed into a CARP solution as shown in Section 2.2. The CARP solution is feasible but it may contain additional edges that are not needed to form feasible routes and could be substituted by short cuts. This is done by successively checking each route: first we try to exchange required edges of the considered route with corresponding additional (and thus non-required) edges of other routes. Hereby, feasibility of the routes has to be taken into account. Now inside this route all non-required edges are deleted. The remaining subset of required edges of this route has to be reconnected with minimal additional time. If the subset is connected, the resulting problem is a CPP. Otherwise, it is a rural postman problem. The RFCS algorithm does a kind of local enumeration: for each pair of cycles that were connected with additional edges each possible connection of the included required arcs that again leads to a cycle is tried and the best one is chosen.

With this reoptimization procedure it is clear that the improvement in the CARP does not exactly correspond to the attained CMST improvement.

## 2.7. Side constraints and additional objective functions

In the algorithm some side constraints can be considered:

- Edges can be fixed to a specific center. This can be necessary if, e.g., a narrow street needs to be

served by a special vehicle, or a vehicle must pass a load station. By this means, also a fictive (“dummy”) path can be fixed if the start and end node of a route are different. With the fixing of an edge also the CMST cycle node containing this edge is fixed to the respective center.

- Cycles in the CARP graph can be made “by hand” and fixed. Each partitioning then uses the specified cycle(s). Directed arcs can be included in “feasible” cycles, such that each cycle can be traversed in one direction and does not contain arcs with contrary direction. Additionally, such cycles can be fixed to a specific center.
- Time windows can be included in a basic form where, e.g., each route can be divided into an earlier, high priority part and a second, low priority part or by interpreting the vehicles as time periods and fixing edges with corresponding time limits.

Following Zhu (1989) we consider two additional objective functions:

The first objective is to balance the capacity utilization, especially of demand capacity. It is supposed that an unjust assignment of work may lead to discontent among the staff. The average utilization share AU is given as the sum of demand weights of all edges divided by the sum of demand capacity of all vehicles:

$$AU = \sum_{(i,j) \in A} d_{ij} / \sum_{m=1}^M D_m.$$

The utilization share  $U_m$  of vehicle  $m$  is the sum of demand weights of all edges that are served by vehicle  $m$  in route  $R_m$  divided by the capacity of vehicle  $m$ :

$$U_m = \sum_{(i,j) \in R_m} d_{ij} / D_m.$$

The objective is to minimize the utilization deviation DU

$$DU = \sqrt{\frac{1}{M} \cdot \sum_{m=1}^M (U_m - AU)^2}.$$

With a weight factor  $f \in [0, 1]$  the minimization of DU can be embedded in the objective function

of the CARP which becomes: minimize  $f \cdot$  (total time of additional edges) +  $(1 - f) \cdot$  DU. This objective function can be transformed into the CMST context and thus used in the improvement procedure.

The second objective is to serve the edges according to their priority. Assume the existence of several priority classes with large priority weights  $p_{ij}$  for high priority edges  $(i, j)$  and small weights for low priority edges. Let  $st_{ij}$  be the time when the serving of edge  $(i, j)$  is started. The objective is to minimize the priority function  $PF = \sum_{(i,j) \in A} p_{ij} \cdot st_{ij}$ .

For the computation of the start times  $st_{ij}$  the routes  $R_m$  of the  $M$  vehicles specifying the sequence of edges are needed. Therefore, it is impossible to integrate the minimization of PF into the CMST problem where only the assignment of edges to vehicles but no sequencing is done. Thus, the minimization of PF is a subordinate objective being part only of the route optimization procedure. The algorithm of Zhu does not allow for raising the total time of additional arcs or the utilization deviation in order to lower PF.

### 2.8. Solution quality

As we have seen each feasible CMST solution corresponds to a feasible CARP solution. However, it is not assured that each feasible solution of the CARP can be obtained from a CMST solution. This is due to the fact that a CMST cycle node is completely assigned to one center node. But capacity constraints could force the splitting of the cycle to obtain the optimal solution. Thus, it is reasonable to build cycles as small as possible. Also, each partitioning should be tried. There are four reasons for the potential suboptimality of the CARP solution:

- The Euler arcs that are fixed are not part of an optimal solution.
- The partitioning of the Euler graph into cycles is not optimal.
- The CARP optimum needs the splitting of a cycle.
- The solution of the CMST is suboptimal.

### 3. Real-world application – computational results

In this section we consider two real-world benchmark problems dealing with planning routes for winter gritting in the areas of Königstein and Wennigsen. Furthermore, some comments on benchmark instances from the literature are given. Our algorithm is implemented in FORTRAN and was run on a 486 processor with coprocessor with 66 MHz. Within the solution approach for the CMST we used the metastrategies simulated annealing and tabu search with their respective parameters:

<b>SA</b> $t$	simulated annealing with start temperature $t$
<b>STS</b> $s$	static tabu search with tabu list length $s$
<b>CSM</b> $c_1 c_2 c_3$	cancellation sequence method with length of the active tabu list $c_1$ , duration of the tabu status $c_2$ , and length of the tabu buffer $c_3$
<b>REM</b> $r$	reverse elimination method with length of the running list $r$

For CSM and REM we assured the length of the active tabu list  $c_1$  and the length of the running list  $r$  to be greater than the iteration number in order to prevent the loss of information by cutting these lists (i.e. all attributes representing yet performed moves are stored in a respective way, see Amberg et al. (1996) for details).

For each problem several partitionings of the Euler graph into a cycle node graph are derived. We refer to them as CMST partitionings or CMST instances. For each CMST instance, if a feasible solution is found, also a CARP solution is derived.

Tables 1 and 2 give the results of our computations for both problems. The first row gives the result of our algorithm without applying any improvement procedure. In the subsequent rows the metastrategies with their parameter values are investigated. The notation of the table is as follows:

*Best CARP solution:* gives the length of additional edges (including Euler edges) of the best CARP solution found with the respective method (considering all investigated CMST instances).

*Number of best CMST solutions:* gives the number of times the best objective function value of any CMST instance is found (as a reference solution we take the overall best CMST solution found by any of the methods considered).

*Average CMST improvement:* gives the average improvement (in %) for the objective function values of the CMST instances obtained with the respective method.

*Average time of last improvement:* gives the necessary average CPU time used by any metastrategy to find the best CMST solution (over the complete set of CMST instances) within the given time limit.

*Average iteration number:* gives the average number of iterations performed within the given time limit (again over the complete set of CMST instances).

The first problem considers the area of Königstein (for more details see Zhu (1989) or Durth and Hanke (1984)). Six vehicles are stationed in the same depot node but have different time and demand capacities. One vehicle is fixed to pass a load station and obtains higher demand capacity. One edge represents a narrow street and is fixed to a vehicle which is small enough to serve it. The CARP graph consists of 65 nodes and 94 edges. For the Euler graph 30 Euler edges with a total length of 59.5 km are added. Travel time is assumed to be proportional to distance and thus the objective is to minimize the total distance of additional edges. Due to capacity constraints at least five routes are needed. Thus, in a feasible solution ten edges have to be incident with the depot node. This raises the lower bound for the length of additional edges in a feasible solution from 59.5 to 68.5 km (cf. Golden and Wong, 1981). (A careful consideration of the given CARP instance reveals that the lower bound may be raised to 76.5 km.) As the number of possible partitionings is considerably large (over 50 000) we decided to select a number of 30 different partitionings at random. Each partitioning leads to a specific CMST instance, each having six centers and about 40 non-center nodes. For each instance the time limit for the improvement procedure is 150 seconds. The best solution that we found has five routes and a total length of additional edges of

Table 1  
Königstein

Metastrategy with parameters	Best CARP-solution (km)	Number of best CMST solutions	Average CMST improvement (%)	Average time of last improvement (s)	Average iteration number
None	92.5	0	0.00	—	—
SA 10	79.5	2	11.35	16	94673
SA 20	79.5	3	12.91	18	94323
SA 40	79.5	11	16.71	35	96687
SA 80	79.5	14	21.84	53	97502
SA 120	79.5	4	16.97	72	98469
STS 7	79.5	11	20.29	45	849
STS 10	79.5	13	23.19	35	862
STS 12	79.5	15	24.45	49	852
STS 15	79.5	21	24.84	56	869
STS 20	79.5	16	24.58	47	888
STS 25	79.5	14	24.52	57	876
STS 30	79.5	9	22.37	57	891
CSM 1400 7 1	79.5	6	14.69	37	840
CSM 1400 10 1	79.5	6	14.94	40	842
CSM 1400 15 1	79.5	16	21.29	63	858
CSM 1400 20 1	79.5	11	19.31	47	851
CSM 1400 7 2	79.5	10	18.10	46	842
CSM 1400 10 2	79.5	10	17.02	35	846
CSM 1400 15 2	79.5	9	18.70	47	834
CSM 1400 20 2	79.5	12	18.93	46	849
CSM 1400 7 4	79.5	13	19.92	44	847
CSM 1400 10 4	79.5	12	19.20	51	833
CSM 1400 15 4	79.5	16	20.76	61	838
CSM 1400 20 4	79.5	13	19.85	39	839
REM 1500	79.5	10	17.50	51	797

Table 2  
Wennigsen

Metastrategy with parameters	Best CARP solution (km)	Number of best CMST solutions	Average CMST improvement (%)	Average time of last improvement (s)	Average iteration number
None	114.2	3	0.00	—	—
SA 20	114.2	9	3.54	0	180039
SA 40	114.2	9	3.54	0	180462
SA 80	114.2	9	3.54	0	181112
SA 120	102.4	13	9.81	7	182444
STS 7	114.2	9	3.54	0	2551
STS 10	102.4	15	12.40	7	2556
STS 15	102.4	17	14.38	8	2440
CSM 4700 7 2	114.2	9	3.54	0	2524
CSM 4700 10 2	102.4	10	4.94	5	2524
CSM 4700 15 2	102.4	11	6.34	11	2508
REM 2700	102.4	12	8.20	15	1835

79.5 km. The best solution found by Zhu (1989) had six routes with 83.5 km total additional edge length. The computational results are shown in Table 1.

Each metastrategy yields an improvement of about 14% over the initial CARP solution showing the usefulness of the improvement procedures. Each method reaches the best known CARP

solution having a total length of 79.5 km. To have a more indepth view at the results, we investigate the CMST instances themselves that were derived from the CARP. Here we get some insight into the methods with respect to their comparative behavior. For this purpose the overall best CMST solution obtained by any of the methods applied in the test is taken as a reference solution, i.e., as upper bound (UB).

The number of best CMST solutions (upper bounds) reached is greatly varying for the SA methods from about 7–48% (i.e., between 2 and 14 out of 30 instances). Average improvements of up to 22% are obtained. In addition, the average deviation from the UBs varies from 26.97% to 11.43% (not directly shown in the table). The CSM results show that the combination of small parameter values for tabu duration  $c_2$  and tabu buffer  $c_3$  performs poorly reaching only 20.69% of the upper bounds (again in absolute values in the table) and about 22% average deviation from these bounds (see results for CSM 1400 7 1 and CSM 1400 10 1). The best CSM methods are CSM 1400 15 1 and CSM 1400 15 4 both reaching 55.17% of the upper bounds with 12.79% and 13.25% average deviation. STS outperforms CSM and SA with respect to the number of best solutions obtained as well as the average CMST improvement having values of up to 21 best solutions and an average improvement of up to 24.84% (see especially values for  $s$  within 12 and 25). The overall best behaviour is shown by STS 15. STS results are best in a mean parameter area (with a large number of different parameter values giving excellent results) being worst for a short tabu list length of 7. Thus, a longer tabu list length should be preferred. REM with 34.48% of the upper bounds with a deviation of 18.1% and an average improvement of 17.5% is not the worst result but far from the best methods. Nevertheless, it could be a good choice if there is no time to experiment with different parameter settings. Note, however, that these conclusions are drawn from a quite small testbed only and need not be valid in general (compare e.g., Amberg et al., 1996).

Nevertheless, all tested metastrategies reach the best known CARP solution – independent from the used parameter values but always throughout the same partitioning of cycle nodes and only

through this one. Thus, it does not seem to be of greatest importance to choose an “optimal” parameter constellation as long as it is “good”. Instead, closest attention should be given to the choice of the partitioning. Therefore, we propose to use the REM because it needs no parameters giving way to multiplying the number of partitionings exploited. Furthermore, it should be tried to investigate a set of partitionings representative to the total set which probably would be too numerous to enumerate. At least it has to be assured that no partitioning is chosen twice which might be the case with random choice as in the considered algorithm. Dominated partitionings e.g., containing cycles that could be split in smaller ones can be dropped, bounding rules could be used to tighten the solution space.

The fixed running time is exploited with a share of 10% up to 50% as derived from the average time of last improvement, and each strategy has a maximal time of last improvement near to the running time boundary. Thus, we expect further improvements when raising the CPU time.

In the second problem, concerning the area of Wennigsen six vehicles are stationed in two depot nodes (cf. Zhu, 1992). The CARP network has 48 nodes and 55 required arcs. The total length of the 20 Euler arcs is 64.6 km. The matching lower bound of Golden and Wong (1981) for additional arc length is 67.6 km. 17 feasible CMST instances with six center nodes and about 20 non-center nodes are derived. The time limit again is 150 seconds. The overall best solution has 102.4 km in six routes. Table 2 shows the details.

SA procedures with low start temperature fail to find the best CARP solution, they do not even improve the initial solution. Regarding the CMST results the best method is STS 15 reaching 100% of the CMST upper bounds. The average time of last improvement is at about 10% of the running time. Thus, further improvements are less likely than in the Königstein example.

In a modified version of the Wennigsen problem we eliminated four of the required arcs resulting in 51 required arcs, 20 Euler arcs of total length 71.9 km, a matching lower bound of 74.9 km and an overall best solution of 104.7 km. Our computations show similar results as in the previous case.

The low temperature SA methods fail, SA with higher temperature (SA 120) and STS 15 perform best. An important question for judging the efficiency of solving the CARP via the CMST is as follows: when our improvement procedures obtain a reduction of the total length of additional edges compared to the initial CMST solution, what is the change with respect to the underlying CARP solution? Our computations showed that for the Königstein data set the ratio of the CARP improvement (concerning additional edge length) and the CMST improvement for each metastrategy is in between 87.5% and 100%, and for the Wennigsen data set we obtained a ratio of 100% in all cases. Thus we can deduce that reducing the CMST cost strongly influences the edge length of the CARP solution and that it seems to be a reasonable approach to use the CMST for solving the CARP.

With respect to a detailed investigation other data sets from the literature seem to be fairly small and it might be questionable whether our approach is reasonable in those cases. For instance, Hertz et al. (1997) in a very recent working paper investigate the 23 so-called DeArmon instances as well as randomly generated instances. The first set contains 20 instances with up to 13 nodes and three instances with up to 27 nodes; the randomly generated instances have up to 19 nodes. We have applied our approach to these instances and compared them to the specialized tabu search algorithm of Hertz et al. (1997). As our average results are slightly worse we carefully investigated those instances where Hertz et al. (1997) obtained better solutions. It turned out that due to the small number of nodes and the respective structure of the instances, the diversity of the CMST partitionings is not sufficiently large and we were not able to eliminate the additional Euler edges in some cases. Our approach behaves superior when the capacity of a route subsumes a large number of small cycles with a comparatively small number of non-required arcs.

#### 4. Conclusions

In this paper we have considered a specific arc routing problem that appears in real-world appli-

cations, e.g., in winter gritting. A route-first-cluster-second approach incorporating the treatment of a modified capacitated minimum spanning tree problem as well as modern heuristic concepts have been proposed. As a side-effect we have also introduced the arc-constrained CMST which is a further generalization of the well-known CMST.

The heuristic for the arc routing problem is capable of yielding good solutions but the running time is significant. Therefore, a more systematic choice of the partitioning into cycle nodes, dominance criteria and bounding rules could be applied, e.g., within a branch and bound procedure. Furthermore, the possibility of a parallel exploration of different partitionings could be considered or even a distinct, parallel optimization of local areas of the network as one part inside the overall optimization. This could be of even greater use when dealing with a rural postman problem with multiple depots which the above algorithm is able to handle with some specific adaptation. Moreover, we propose the reverse elimination method as an appropriate metastrategy when running time is restrictive as no parameter settings are required.

#### Acknowledgements

The helpful comments of the two referees, which helped to improve the presentation of the paper, are greatly appreciated.

#### References

- Aarts, E., Korst J., 1989. Simulated annealing and Boltzman machines. Wiley, Chichester.
- Amberg, A., Domschke, W., Voß, S., 1996. Capacitated minimum spanning trees: Algorithms using intelligent search. *Combinatorial Optimization: Theory and Practice* 1, 9–39.
- Assad, A.A., Golden, B.L., 1995. Arc routing methods and applications. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (Eds.), *Network Routing. Handbooks in OR/MS*, vol. 8, Elsevier, Amsterdam, pp. 375–483.
- Benavent, E., Campos, V., Corberan, A., Mota, E., 1990. The CARP – a heuristic algorithm. *Questio* 14 (1–3), 107–122.
- Bodin, L.D., 1975. A taxonomic structure for vehicle routing and scheduling problems. *Computers and Urban Society* 1, 11–29.

- Chapleau, L., Ferland, J.A., Lapalme, G., Rousseau, J.M., 1984. A parallel insert method for the capacitated arc routing problem. *Operations Research Letters* 3, 95–99.
- Christofides, N., 1973. The optimum traversal of a graph. *OMEGA* 1, 719–732.
- Clarke, G., Wright, J.W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* 12, 568–581.
- Dijkstra, E.W., 1959. A note on two problems in connection with graphs. *Numerische Mathematik* 1, 269–271.
- Durth, W., Hanke, H., 1984. Optimierte Routenplanung für den Winterdienst der Straßenmeistereien Reichelsheim, Alsfeld und Königstein. *Untersuchungen für das Hessische Landesamt für Straßenbau, Darmstadt*.
- Edmonds, J., Johnson, E.L., 1973. Matching, Euler tours and the Chinese postman. *Mathematical Programming* 5, 88–124.
- Esau, L.R., Williams, K.C., 1966. On teleprocessing system design. *IBM Systems Journal* 5, 142–147.
- Gavish, B., 1983. Formulations and algorithms for the capacitated minimal directed tree problem. *Journal of the Association for Computing Machinery* 30, 118–132.
- Gavish, B., Graves, S., 1979. The travelling salesman problem and related problems. Working paper, Vanderbilt University, Nashville, TN.
- Geppert, B., 1986. Tourenplanung bei der innerstädtischen Hausmüllentsorgung. PhD. Thesis, Technical University of Karlsruhe.
- Glover, F., 1989. Tabu search – part I. *ORSA Journal on Computing* 1, 190–206.
- Glover, F., 1990. Tabu search – part II. *ORSA Journal on Computing* 2, 4–32.
- Golden, B.L., Wong, R.T., 1981. Capacitated arc routing problems. *Networks* 11, 305–315.
- Golden, B.L., DeArmon, J.S., Baker, E.K., 1983. Computational experiments with algorithms for a class of routing problems. *Computers and Operations Research* 10, 47–59.
- Gouveia, L., 1995. A  $2n$ -constraint formulation for the capacitated minimal spanning tree problem. *Operations Research* 43, 130–141.
- Gouveia, L., Lopes, M.J., 1995. Using generalized capacitated trees for designing the topology of local access networks. In: *3rd International Conference on Telecommunication Systems*, Nashville, pp. 320–327.
- Gouveia, L., Martins, P., 1996. The capacitated minimal spanning tree problem: an experiment with a hop-indexed model. Working paper, Faculdade de Ciências da Universidade de Lisboa.
- Hall, L., 1996. Experience with a cutting plane algorithm for the capacitated spanning tree problem. *INFORMS Journal on Computing* 8, 219–234.
- Hertz, A., Laporte, G., Mittaz, M., 1997. A tabu search heuristic for the capacitated arc routing problem. Working Paper, C.R.T., University of Montreal.
- Kershenbaum, A., Boorstyn, R.R., Oppenheim, R., 1980. Second-order greedy algorithms for centralized teleprocessing network design. *IEEE Transactions on Communications* 28, 1835–1838.
- Kruskal, J.B., Jr, 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society* 7, 48–50.
- Kwan, M.-K., 1962. Graphic programming using odd or even points. *Chinese Mathematics* 1, 273–277.
- Liebling, T.M., 1970. *Graphentheorie in Planungs- und Tourenproblemen am Beispiel des städtischen Straßendienstes*. Springer, Berlin, Heidelberg, New York.
- Liebman, J.C., Male, J.W., 1974. Routing of solid waste collection vehicles. Final report. Appendix B: A heuristic solution to the M-postmen's problem. Illinois University, 444 Urbana, US Environmental Protection Agency Report No. 670/2-74-036b.
- Papadimitriou, C.H., 1976. On the complexity of edge traversing. *Journal of the Association for Computing Machinery* 23, 544–554.
- Papadimitriou, C.H., 1978. The complexity of the capacitated tree problem. *Networks* 8, 217–230.
- Pearn, W.L., 1989. Approximate solutions for the capacitated arc routing problem. *Computers and Operations Research* 16, 589–611.
- Pearn, W.L., 1991. Augment-insert algorithms for the capacitated arc routing problem. *Computers and Operations Research* 18, 189–198.
- Prim, R.C., 1957. Shortest connection networks and some generalizations. *Bell Syst. Techn. Journal* 36, 1389–1401.
- Pütz, W., 1979. *Numerische Untersuchungen zum Chinesischen Postbotenproblem*. Diplomarbeit am Mathematischen Institut der Universität Köln.
- Sharaiha, Y.M., Gendreau, M., Laporte, G., Osman, I.H., 1997. A tabu search algorithm for the capacitated shortest spanning tree problem. *Networks* 29, 161–171.
- Stern, H.I. Dror, M., 1979. Routing electric meter readers. *Computers and Operations Research* 6, 209–223.
- Zhu, P., 1989. Ein flexibles Verfahren zur Lösung kantenorientierter Tourenplanungsprobleme im Straßenbetriebsdienst. PhD. dissertation, Technische Hochschule Darmstadt.
- Zhu, P., 1992. Personal Communication.