Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computers & Operations Research 33 (2006) 894–909

computers & operations research

www.elsevier.com/locate/cor

A tabu search heuristic for the truck and trailer routing problem

Stephan Scheuerer[*]

School of Business, University of Regensburg, Universitätsstraße 31, D-93053 Regensburg, Germany

Available online 29 September 2004

Abstract

Two new construction heuristics and a tabu search heuristic are presented for the truck and trailer routing problem, a variant of the vehicle routing problem. Computational results indicate that the heuristics are competitive to the existing approaches. The tabu search algorithm obtained better solutions for each of 21 benchmark problems.

Many real-life vehicle routing applications include the use of trailers. Whenever a truck and a trailer can be treated as a single vehicle, that means the trailer is never uncoupled, a normal vehicle routing problem can be solved. However, as is the case in the truck and trailer routing problem, customers may exist that are not reachable by trailer. For this reason, the trailer has to be uncoupled and left behind at a parking place and has to be again picked up at a later point in time on the route. Besides routing, the problem therefore includes also the decision of determining the best parking place and the number of times a trailer should be uncoupled. The purpose of this article is to introduce two simple, but efficient, construction heuristics for this problem and to present a tabu search heuristic with a variable number of sub-tours for further improvement.

© 2004 Published by Elsevier Ltd.

Keywords: Vehicle routing problem; Truck and trailer routing problem; Tabu search heuristic

1. Introduction

The *vehicle routing problem* (VRP) is one of the most studied problems in the field of operations research. It consists of finding least cost routes for a set of homogeneous vehicles located at a depot to geographically scattered customers. Each customer has a known demand and service duration. The routes

[*] Tel.: +49-941-943-2728; fax: +49-941-943-4979.

E-mail address: stephan.scheuerer@wiwi.uni-regensburg.de (S. Scheuerer).

0305-0548/$ - see front matter © 2004 Published by Elsevier Ltd.
doi:10.1016/j.cor.2004.08.002

have to be designed such that each customer is visited only once by exactly one vehicle, each vehicle route starts and ends at the depot and the total capacity of a vehicle may not be exceeded. The VRP is computationally hard to solve and is usually tackled by heuristic approaches, see for example Toth and Vigo [1], Laporte et al. [2] and Cordeau et al. [3].

A commonly neglected aspect in vehicle routing is the usage of trailers. The *truck and trailer routing problem* (TTRP) concentrates on this fact. The TTRP can be defined on a graph $G=(V, A)$, with vertex set $V=\{v_0, v_1, \dots, v_n\}$ and arc set $A=\{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$. The vertex v_0 represents the depot where m trucks and b trailers ($b \leqslant m$) are based. The trucks and trailers are considered homogeneous with capacity K and R, respectively. A truck plus a trailer is called a *complete vehicle*, and a vehicle without a trailer is called a *pure truck*. Each vertex of $V \setminus \{v_0\}$ corresponds to a customer v_i with a non-negative demand q_i and a customer type t_i. The TTRP considers two forms of customer types: a customer who is accessible with or without a trailer is called a *vehicle customer* (VC) and one who is only accessible without a trailer is called a *truck customer* (TC).

If only TC customers were present, the problem could be solved as a normal VRP, neglecting the usage of trailers. On the contrary, if only VC customers were present, the problem could also be solved as a VRP (with heterogeneous vehicles if $b<m$), as there is no need for uncoupling the trailers. However, in real-life situations, both types of customers may exist. For instance, consider TC customers as customers in either a mountainous or an inner-city area or where access is restricted due to the lack of a turning point for the vehicle with an assigned trailer. This fact makes it necessary to consider uncoupling a complete vehicle's trailer at a parking place. In this case a complete vehicle may start from the depot with the assigned trailer, servicing VC customers only. The section of the route where a trailer is assigned will be called the *main-tour* of the *complete vehicle route* (CVR), starting and ending at the depot. In a CVR it is now possible to uncouple the trailer at a parking place and to start a *sub-tour* from this point. On a sub-tour, both types of customers are accessible. Sub-tours have to be designed such that each starts and ends at the same parking place where the uncoupling took place in order to reattach the trailer. Please note that there is no restriction on the number of sub-tours for a CVR and that a parking place can be used for more than one sub-tour of the CVR. A CVR therefore consists of exactly one (possibly empty) main-tour and none, one or several sub-tours. For a complete vehicle, the sum of all demands collected on the CVR may not exceed the vehicle's total capacity, that is $K+R$. Also, the truck capacity K may not be exceeded on any sub-tour. Thereby it is assumed that shifting demand loads between the truck and the trailer is possible at the parking places. In contrast to a complete vehicle, a pure truck is always able to service both types of customers. The corresponding route is called a *pure truck route* (PTR), starting and ending at the depot. It does not possess a sub-tour and is limited by the truck capacity K. Note that Chao [4] explicitly defined a third type of route in his model, a so-called *pure vehicle route*. This is a special case of a CVR without sub-tours.

Following the model of Chao [4] for the TTRP, every VC customer and also the depot can be selected as a trailer-parking place. However, it is not possible to use a VC customer as a parking place in different main-tours, as this VC customer is to be considered for service in exactly one main-tour. This, of course, does not apply to the depot that is part of every route.

Regarding the tours, the starting and ending node is called the *root* of the tour. For main-tours of a CVR and for a PTR, the root is always the depot; for sub-tours it can be the depot or a VC customer serving as the parking place for the trailer. Note that the term *tour* is used to denote a single, simple roundtrip of a vehicle, whereas a *route* may contain several tours. For every route there may be a restriction on total route length.

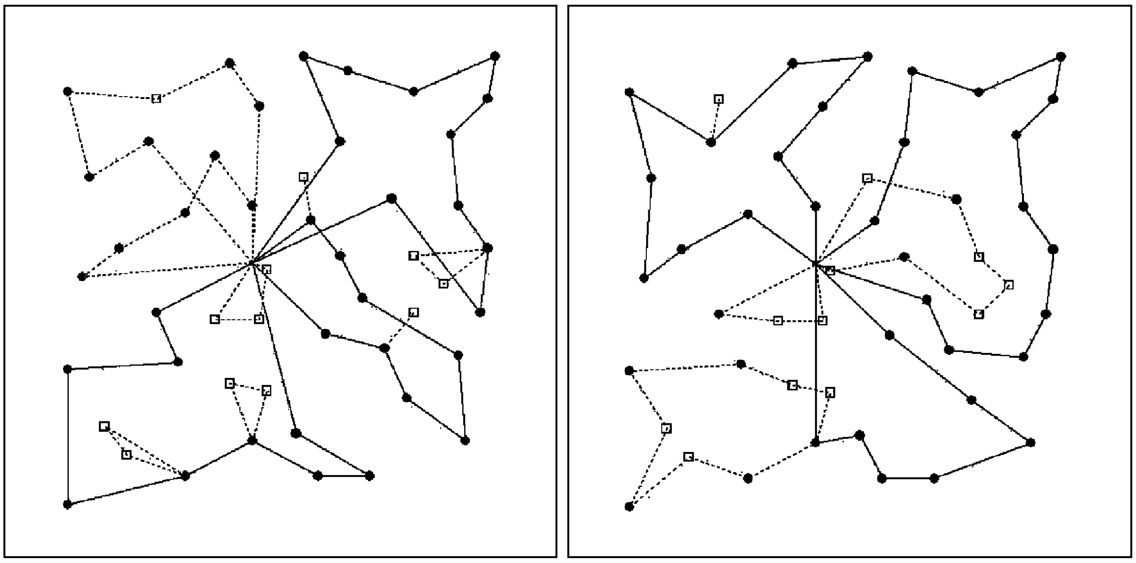

Fig. 1. Two feasible solutions to test problem 1.

The objective of the TTRP consists of finding routes such that the total travel costs are minimized and all the constraints described above are satisfied. Therefore a cost matrix $C = (c_{ij})$ is associated with the arc set, representing non-negative travel times or distances from v_i to v_j. Thereby, the TTRP additionally includes the problem of finding the optimal number of sub-tours and the location of the parking places for complete vehicle routes.

Fig. 1 shows two feasible solutions to test problem 1, created by Chao [4], consisting of 38 VC and 12 TC customers, three CVRs and two PTRs. TC customers are marked by white squares and VC customers by black dots. Solid lines specify main-tours and dashed lines specify tours without trailers, that is sub-tours or pure truck routes. The left solution was created by the T-Cluster heuristic described in Section 3 and the right picture shows the best known solution to this test problem, with routing costs of 651.87 and 564.68, respectively.

It is worth noting that the TTRP model in this paper corresponds to the model of Chao [4] without any modifications. Especially the objective function considers total distances and does not include any further cost components like fixed costs, costs for shifting demand, or costs for coupling or uncoupling of trailers. Also the routing costs do not differentiate between tours with and without trailers.

The rest of the paper is organized as follows: Section 2 gives an overview of solution approaches to the TTRP and related problems. In Section 3 two new construction heuristics are presented, followed by a description of a tabu search approach for the TTRP in Section 4. Section 5 shows the computational results of the algorithms, including details on the parameter tuning for the tabu search heuristic. Conclusions are given in Section 6.

2. Literature review

While the VRP and some of its variants have received a lot of attention, the TTRP and closely related problems have seldom been studied.

Semet and Taillard [5] consider a real-life VRP that includes the use of trailers under accessibility restrictions. Their problem is different from the TTRP, as they do not allow a VC customer (called ‘trailer-store’ in their approach) to be served in a sub-tour and as their problem includes further restrictions like time windows and vehicle-dependent variable costs. They propose a construction heuristic based on clustering principles and a tabu search method for this problem.

Semet [6] models a related problem called the ‘partial accessibility constrained VRP’. The model is quite similar to the TTRP, except that the number of sub-tours to a parking place is restricted to a maximum of one and that the depot can be visited only once in a route. The model further assumes that all trucks are used and that the number of trailers must be determined. He proposes a heuristic based on the generalized assignment method of Fisher and Jaikumar [7] for the VRP and presents a branch-and-bound algorithm with Lagrangian relaxation for the trailer assignment problem.

Gerdessen [8] conducted a study on the ‘VRP with trailers’. Gerdessen’s model differs from the TTRP in that all customers have unit demand, customers were assigned manoeuvring costs instead of customer types, each customer location can be used as a parking place and each trailer is parked exactly once. Furthermore, the model considers different driving speeds of vehicles with and without trailers. Four construction heuristics to this problem were presented, including descent improvement phases.

The model for the TTRP used in this article was first proposed by Chao [4]. He developed a construction heuristic and a tabu search method for this problem. The construction heuristic is based on a generalized assignment approach for vehicle routing [7], followed by a descent improvement phase. During the construction phase, VC customers are forced onto main-tours. The tabu search method includes intermediate descent phases and uses a dynamically adjusted threshold for the allowed maximal deterioration of the objective function value. For a detailed description of the heuristics, the reader is referred to the original article. Two details should however be mentioned: first, the root node of a sub-tour may only change during a descent phase of the tabu search, not allowing for deteriorations in the objective function value; second, the number of sub-tours for each CVR is determined in the construction heuristic and is not allowed to increase anymore during the entire tabu search. It is supposed that these are two major points for possible improvements. Note that even if Chao’s formal model for the TTRP includes a restriction on total route length, it remains unclear how his algorithm would handle such a constraint.

3. Construction heuristics

In this section two construction heuristics for the TTRP, called *T-Cluster* and *T-Sweep* are presented. Both heuristics can be easily modified to include further real-life side constraints, for instance, multiple use of vehicles or additional cost components. For an overview on VRP-construction heuristics, see Cordeau et al. [3], Laporte and Semet [9] and Van Breedam [10], for example. T-Cluster can be considered as a cluster-based sequential insertion procedure, where routes are constructed one-by-one up to full vehicle utilization. A new route is initialized with the unrouted customer u farthest away from the depot and the unused vehicle having maximum total capacity. Thereby, complete vehicles are always preferred over pure trucks. In case of a complete vehicle, if the seed customer is a VC customer, then it is inserted into the main-tour. On the other hand it is inserted into a new sub-tour to the depot, if it is a TC customer. The next customer for insertion into the route is then selected from the unrouted customers minimizing $e(k) = c_{ku} + c_{kf} - \pi c_{0k}$, where k specifies the customer under consideration, f the nearest customer already routed in this route, and 0 the depot. For customer k the formulae can be explained as follows:

the first term c_{ku} measures the distance to the seed customer u of this route. This distance should be kept at a minimum to ensure compact routes. The second term c_{kf} ensures that the customer is close to the 'border' of the current route. This proves helpful in instances with clustered customers, where a route should preferably serve all customers of the group before spreading to another group of customers. Finally, the third term πc_{0k} can be considered as a diversification term. Varying the parameter π will result in different selection strategies: the higher the value of π, the higher the tendency for a selection of a customer located far away from the depot.

Having specified the next customer k, a feasible insertion into the route is evaluated via cheapest insertion. In case of a CVR, VC customers are forced into the main-tour and TC customers into a new or existing sub-tour. Every VC customer on the main-tour and the depot can thereby be selected as a trailer-parking place for a new sub-tour. If the insertion of customer k would result in a violation of total vehicle capacity or maximum route length, a new route is initialized and the customer k remains unrouted. Only the last route (and its sub-tours in case of a CVR) are allowed to become infeasible if no more vehicles are available.

After the insertion of a customer into an existing tour, a descent sub-tour root refining is performed and the concerned tour is post-optimized by a 2-opt [11] and Or-opt [12] intra-tour improvement heuristic.

The sub-tour root refining tries to find a better root node for CVR sub-tours. Whenever a VC customer is inserted into the main-tour, the sub-tour root refining is applied to every sub-tour of the CVR to ensure the best possible root for all its sub-tours. It is, however, only applied to the affected sub-tour, when the inserted customer is of type TC. The sub-tour root refining proceeds as follows: first, the two edges that link the sub-tour to the main-tour are removed, and a direct connection, between the resulting first and last customer of the sub-tour, is inserted. The resulting structure is called a sub-tour *loop*. Then the depot and every VC customer of the main-tour (except the former root) are evaluated as the new root node to the sub-tour. The insertion position of the new parking place thereby is determined by cheapest insertion before and after every customer on the sub-tour loop. The best alternative root is chosen if it improves upon total distance traveled. A modified sub-tour is post-optimized by a 2-opt and Or-opt improvement heuristic.

T-Sweep is a heuristic based on the classical sweep algorithm commonly attributed to Gillett and Miller [13]. It suits best to planar instances with a centrally located depot. The method constructs feasible routes by rotating a ray centerd at the depot and gradually including customers in a vehicle route. A new route is initialized with the unused vehicle having maximum total vehicle capacity, whenever total vehicle capacity or route length constraint is attained. If no more unused vehicles are available, the last route is allowed to become infeasible. Evaluation of customer insertion, routing, and post-optimization is done the same way as in the T-Cluster heuristic.

4. Tabu search algorithm

Tabu search is a local search-based metaheuristic that has been successfully applied to a widespread variety of combinatorial optimization problems. Its use was first proposed by Glover [14] in 1986. In contrast to descent methods, the basic tabu search guides the search to avoid getting trapped in local optima by moving at each iteration from a solution s to the best possible neighbor solution $\bar{s}$, even if it causes a deterioration in the objective value. To prevent the search from cycling, attributes of recently visited solutions are memorized in a *tabu list* for a number of iterations (*tabu duration*). Neighbor solutions of s

containing such an attribute are considered temporarily *tabu* or forbidden, unless they fulfill a so-called *aspiration criterion*. For more details on tabu search refer to Glover and Laguna [15,16], Gendreau [17], and in the context of the VRP to Cordeau and Laporte [18]. For an overview on metaheuristics see Glover and Kochenberger [19] and Osman and Kelly [20], for example. In the following, the components of the new tabu search heuristic for the TTRP will be described.

4.1. Evaluation function

In this tabu search heuristic, intermediate infeasible solutions are allowed during the search and infeasibility is controlled by a shifting penalty approach [21]. A solution s is then evaluated by the objective function $f(s) := c(s) + \alpha q(s)$ with $\alpha \geqslant 0$. Total distance $c(s)$ is calculated as the sum of the distances of all tours and total overcapacity $q(s)$ is the sum of the overcapacities of all PTRs and all sub-tours (if demand exceeds K) as well as all CVRs (if total vehicle capacity $K + R$ is exceeded). In a feasible solution $q(s)$ equals zero. The parameter α is used to adjust the penalty term. Following the approach of Cordeau et al. [22], α is initially set to 1 and at each iteration is multiplied by $(1+\delta)$ if the current solution is infeasible $(\delta \geqslant 0)$, but otherwise is divided by $(1+\delta)$. Extreme values for α, however, may not produce the desired effect for guiding the search and therefore α was restricted to vary between 0.01 and 100. Note that even if a route length restriction does not appear in the test instances and therefore is not explicitly presented, the tabu search is able to handle it by adding a second penalty term to the objective function and treating it like the other penalty term, as for example in Cordeau et al. [22].

4.2. Neighborhood

The neighborhood $N(s)$ of a solution s is specified by all possible solutions that can be obtained by applying one of the following transformations to s:

- Shift of consecutive nodes to existing tours or new sub-tours.
- Swap of two subsets of nodes between two existing tours.
- Sub-tour root refining.

Let (T_p, T_q) be a pair of tours not necessarily belonging to different routes, shifting a non-empty subset S_p of consecutive nodes of size $|S_p| \leqslant v_1$ from T_p to T_q is considered. The move is called a *shift move* and the parameter v_1 determines the length of the customer string to be shifted. Setting $v_1 = 1$ means shifting exactly one customer. A shift move may reduce the number of tours in the solution. To allow for new tours to be built up, a shift of the subset S_p into empty PTRs, empty CVR main-tours and into new sub-tours of a CVR route is considered. Note that CVR main-tours and PTRs are considered static in this implementation and therefore can become empty (with respect to the number of customers). In contrast, sub-tours are handled dynamically. They are deleted in case of emptiness and must be created anew if a new sub-tour is built up. This dynamic handling allows for a varying, unlimited number of sub-tours during the search and is a valuable component of the algorithm. However, the evaluation of new sub-tours is restricted to only such cases, where the subset S_p to be moved contains at least one TC customer. Root candidates for a new sub-tour are VC customers on a CVR main-tour or the depot. The current root node of source tour T_p is excluded as a root candidate in case of an intra-route move.

If a non-empty subset S_p of consecutive nodes of size $|S_p| \leqslant v_2$ is shifted from T_p to T_q and *simultaneously* a non-empty subset S_q of consecutive nodes of size $|S_q| \leqslant v_2$ is shifted from T_q to T_p, this exchange procedure is called a *swap move*. Swap moves only appear between non-empty tours and cannot reduce or increase the number of tours. Note that this definition considers only consecutive nodes and can therefore be regarded as a restricted λ-interchange neighborhood [23]. Insertion of consecutive nodes is done by cheapest insertion and evaluating the inversion of the customer string, while keeping the sequence of the customers fixed. After the best move has been performed, a 2-opt and an Or-opt procedure is used to post-optimize modified tours. For both, shift and swap moves, no TC customers are allowed to be inserted into a CVR main-tour and no root node may be moved. However, in contrast to the construction heuristics, VC customers are now allowed to enter sub-tours.

The *root refining* of sub-tours evaluates a reconnection of an existing sub-tour to all other root candidates on its main-tour. The move follows the description in Section 3, unless now also refinings that deteriorate the objective function are accepted. Note that the move can be seen as an intra-route shift move to new sub-tours, where the subset S_p to be moved includes all customers except the root node. The difference to a normal shift move is, however, that the customers may have to be resequenced before evaluating the insertion.

4.3. Neighborhood reduction

To speed up the evaluation process, the neighborhood is further restricted by the parameter h $(h \leqslant n)$. For all shift and swap moves, evaluation of the insertion before, after or to a node u is only performed, if node u is one of the h nearest nodes (including the depot) to the first customer i of the set of consecutive nodes to be inserted. In this case u is called an *h-neighbor* of i. For $h = n$, all other customers and the depot will be examined, otherwise only a subset of the nodes will be considered. Note that in a swap move, the exchange-subset could include the only h-neighbor on this tour. To ensure an insertion position for each subset in a swap move, the remove position of the other subset is always evaluated, regardless of the parameter h.

For a given subset S_p of consecutive nodes on tour T_p, the parameter h is also used to determine the subsets to be evaluated for exchange in a swap move. An exchange with a subset S_q of consecutive nodes on tour T_q is only considered, if the first customer of S_q is an h-neighbor of the first customer in S_p. The reverse relationship is not checked for simplification.

Note that the definition considers only the h-neighbors of the *first* customer in a subset of consecutive nodes. This helps to speed up the search process but requires the definition of a beginning customer for a subset of consecutive nodes. Therefore and to allow for a fast evaluation of the neighborhood, the tours are searched in only one direction. As a consequence, this places a restriction on the number of possible subset combinations in the case of a swap move, where for an h-neighbor only the next customers following the tour direction are evaluated as exchange-subsets.

4.4. Candidate set

Instead of evaluating every possible move at each iteration, the evaluation is restricted to only such moves, where the first customer of a subset S_p of consecutive nodes is in the *candidate set* $M(s)$. The candidate set $M(s)$ is a set of customers randomly selected at each iteration among all customers in the solution s. The size of the candidate set is defined as $|M(s)| = [n/\omega]$ customers and is determined by the parameter ω $(1 \leqslant \omega \leqslant n)$ called *sampling size*. If $\omega = 1$, all customers in s are included and no further restrictions are applied. For simplification and to speed-up the swap-moves, only the first customer of a given subset S_p is examined of being in $M(s)$ and no further examination is performed for the set of possible exchange-subsets to S_p for their beginning customers being in $M(s)$. Comparable to shift and swap moves, a root refining move, for a given sub-tour, is only applied, if the first customer after the root node is included in the candidate set.

4.5. Tabu status and acceptance criterion

Most tabu search heuristics for the VRP restrict the reinsertion of a customer into a tour where it has recently been removed. This requires a fixed tour-reference and can therefore not be used in this approach, because of the dynamic handling of the sub-tours. Identifying a tour by its root node and its route, however, offers a good possibility to (not necessarily identically) refer to a tour, assuming a fixed route-reference for each vehicle. Using only the root to identify a tour would not be as precise and could be too restrictive during the search. For example, consider the depot that is the root of several tours in different routes.

In this implementation, the *tabu criterion* forbids the reinsertion of a customer i into a tour to root node k in route l for the next θ iterations. More precisely, if customer i $(i \in V \setminus \{v_0\})$ is removed from a tour to root node k $(k \in V, k \neq i)$ in route l $(1 \leqslant l \leqslant m)$, the attribute (i, k, l) is added to a tabu list (overwriting similar entries, if necessary). With every attribute in the tabu list is associated the number of the last iteration τ_{ikl} for which it is denoted tabu. Let λ be the iteration counter, the remaining tabu duration of the attribute (i, k, l) for the current iteration can be calculated as $\max\{\tau_{ikl} - \lambda; 0\}$ and is considered zero for all attributes not in the tabu list. A transition from a solution s to a neighbor solution $\bar{s} \in N(s)$ involves moving at least one customer to another tour (regarding the root refining as a special shift move). Each move can then be specified by a set of attributes $B(\bar{s}) = \{(i, k, l)$: customer i is inserted in a tour to root node k in route $l\}$. The tabu duration $\tau(\bar{s})$ of a neighbor solution $\bar{s} \in N(s)$ is determined by calculating the minimum over the remaining tabu durations of all attributes appearing in $B(\bar{s})$. A neighbor solution $\bar{s}$ is considered tabu, if $\tau(\bar{s}) > 0$ and non-tabu, if $\tau(\bar{s}) = 0$ or if it fulfills the *aspiration criterion*. In this implementation, an aspiration criterion is applied that releases the tabu status of a move by setting $\tau(\bar{s}) = -1$, if the move leads to a new best feasible solution. The neighbor solution $\bar{s}$ which has the least value of $\tau(\bar{s})$ is realized, if ambiguous the one with the least deterioration of the evaluation function value.

4.6. Long-term memory

To diversify the search, a *long-term memory* is implemented and a penalty term is added to the evaluation function whenever a move leads to an increase in the current objective function value. Let ρ_{ikl} be the number of times customer i has been inserted into a tour to root k in route l. The average relative frequency $\rho(\bar{s})$ for a move is then computed as the average of the ρ_{ikl} values of all stops involved in the move, divided by the number of iterations performed on the given problem. The penalty term is adjusted by a constant factor γ and is computed as $r(\bar{s}) = \gamma \cdot c(s) \cdot \sqrt{(n \cdot rc \cdot m)} \cdot \rho(\bar{s})$, where n denotes the number of customers, rc the number of root candidates including the depot and m the number of trucks in solution s, respectively. Total distance $c(s)$ of the current solution s and the square root factor are used to adjust the penalty value. The square root factor can be considered as a normalizing factor because the frequency of occurrence of

Table 1
Function TABUSEARCH of the algorithm

```
Input: Initial solution $s$.
Output: Best solution found $s^*$.
  (1) Set all tabu durations and frequency counters to zero. Set $\alpha := 1$.
  (2) While the stopping criteria is not met do
    (a) Determine candidate set $M(s)$ and neighborhood $N(s)$.
    (b) Choose the neighbor solution $\bar{s} \in N(s)$ with minimum tabu duration $\tau(\bar{s})$
        and if ambiguous, with minimum $g(\bar{s})$,
        where $g(\bar{s}) := f(\bar{s}) + r(\bar{s})$ if $f(\bar{s}) \geqslant f(s)$ and $g(\bar{s}) := f(\bar{s})$ otherwise.
    (c) Set $s := \bar{s}$ and update the tabu list and long-term memory.
    (d) Perform a 2-opt and Or-opt post-optimization on modified tours.
    (e) Check for update on the best solution $s^*$.
    (f) Update parameter $\alpha$ and increase the iteration counter $\lambda$.
```

a customer being inserted into a tour to a given root and route decreases with the size of the problem and the number of root candidates. Its use in the context of VRP was first proposed by Taillard [24].

4.7. *Intensification through restarts*

To intensify the search at promising regions of the solution space, restarting the search from the current best solution s^* is considered after a certain number of iterations μ without improving s^*. This strategy is referred to as I_1 and an implementation without restarts as I_0. Experimental studies showed that a continuing restart from the same solution may hinder the search to advance to other parts of the solution space. Therefore a limit of three as the maximum number of restarts from the same solution s^* was set; this was experimentally found to be a good value. Note that the candidate set sampling strategy is an essential component when using a restart strategy, because it forces the search to proceed on different paths, when it is restarted from the same solution.

Furthermore, each CVR can be considered as a difficult routing problem itself. To improve on the structure of a CVR, an application of the basic I_0 tabu search is considered separately to every CVR at each restart in I_1. This advanced restart strategy is denoted as I_2. For an application to a CVR, h is set to infinity and a stopping criterion of μ iterations without improving s^* is used. Relaxing the value of h is done to avoid the risk of an inappropriate setting, which may hinder a CVR from reaching its optimal structure. Note that a feasible CVR with only one tour is already optimal in reference to the route structure. As a feasible solution could always be found for each test problem before applying the first restart, the CVR improvement is restricted to CVRs with at least two tours.

4.8. *Overview of the tabu search heuristic*

Now a description of the heuristic with the I_2 intensification strategy is presented. Table 1 shows the general outline of the basic tabu search, denoted as function TABUSEARCH, as described in Sections 4.1–4.6. Note that the function may not necessarily be applied to the entire problem and that all parameters used for calculations inside the function refer to the problem at hand. For example, for a CVR improvement, the solution s only means the CVR and when calculating the long-term penalty, the parameters n and rc

Table 2
Outline of the tabu search heuristic (I_2 strategy)

```
(1) Generate an initial solution $s$ with a construction heuristic.
        Set the best solution $s^* := s$ and the iteration counter $\lambda := 1$.
(2) While ($\lambda \leqslant \eta$) do
      (a) Apply TABUSEARCH to the entire best solution $s^*$.
      (b) Apply TABUSEARCH to every CVR in $s^*$ with at least two tours.
(3) Apply a descent search with $\omega = 1$ to the entire best solution $s^*$.
```

only refer to the number of customers and root candidates in this CVR. In TABUSEARCH, the search itself is carried out in step 2 until μ iterations without improving s^* have been performed. Parameter μ is temporarily relaxed to infinity, if the maximum number of three restarts on the same solution has been reached and s^* was not improved since then.

In Table 2 the outline of the entire algorithm is presented. The algorithm stops when a fixed number of iterations η has been performed. Because sampling may hinder the search to find a true local optimum, a descent search, using the same neighborhood as the tabu search, but without sampling, is applied to the entire best solution s^* at the end of the algorithm.

5. Computational results

The heuristics were coded in C++ (using Microsoft Visual C++ 6.0) and all tests were performed on a PC with Pentium IV 1.5 GHz processor, rounding costs to two decimal places during the run of the algorithms. As suggested in Cordeau et al. [3], the final solutions' costs were then recalculated with double precision and again rounded to two decimal places for presentation. For this analysis, the 21 test problems for the TTRP created by Chao [4] were used.

5.1. *Results of the construction heuristics*

Both the T-Cluster and the T-Sweep heuristics are designed as multi-start procedures. The T-Cluster heuristic performed best when run for a total of 46 times with different values of π, starting from -1.5 up to +3.0 with an increment of 0.1 at each iteration. T-Sweep was re-started n times, using every customer exactly once as the seed customer for the first route.

A detailed comparison of the two algorithms with the construction heuristic of Chao, with and without descent improvement [4, p. 48], is given in Table 3. These include two or three columns each, specifying total distance $c(s)$, total overcapacity $q(s)$ and — when available — total runtime in seconds T for each of the 21 test problems. No times are available for Chao's heuristics. Total distance of the best known solution s^{**} is given in the outermost right column. The last two rows show the column average (avg) and average relative percentage deviation (ARPD) to the best known solutions for the 21 test problems. Please note that Chao (only) reports the average over 10 runs, while the best solutions produced by T-Cluster and T-Sweep are presented, as this is the actual result that will be forwarded to the tabu search heuristic. Note also that all construction heuristics except Chao's heuristic with descent improvement force VC customers in main-tours of a CVR.

The results indicate that the T-Cluster heuristic dominates the other heuristics in solution quality. It found feasible solutions to all 21 test problems at lowest average routing costs with an ARPD of 15.22.

Table 3
Comparison of construction heuristics

ID	T-Cluster[a]			T-Sweep[b]			Chao constr.[c,e]		Chao descent[c,e]		$c(s^{**})$
	$c(s^*)$	$q(s^*)$	T[d]	$c(s^*)$	$q(s^*)$	T[d]	$c(s^*)$	$q(s^*)$	$c(s^*)$	$q(s^*)$	
1	651.87	0.0	0.33	644.80	0.0	0.30	657.15	9.6	646.02	0.0	564.68
2	697.51	0.0	0.27	722.46	0.0	0.23	739.04	13.9	739.90	0.0	612.75
3	766.25	0.0	0.25	797.65	0.0	0.23	785.54	16.8	774.78	0.0	618.04
4	979.79	0.0	0.45	901.78	26.0	0.56	937.82	26.0	943.47	0.0	798.53
5	1037.50	0.0	0.42	1035.76	32.0	0.55	1108.87	22.9	1130.85	0.0	839.62
6	1173.11	0.0	0.41	1171.99	52.0	0.55	1174.17	32.1	1236.69	0.0	933.26
7	904.77	0.0	1.09	901.14	0.0	1.88	937.31	14.1	906.31	0.0	830.48
8	965.90	0.0	1.00	1005.99	0.0	1.63	1004.45	18.5	971.60	0.0	878.36
9	1081.21	0.0	0.94	1099.88	0.0	1.45	1156.50	45.6	1106.66	0.0	934.47
10	1167.38	0.0	1.83	1150.42	0.0	4.84	1232.10	33.6	1159.78	0.0	1039.07
11	1274.67	0.0	1.94	1288.49	0.0	3.94	1422.41	38.0	1288.74	0.0	1094.11
12	1438.11	0.0	1.69	1443.00	0.0	3.77	1578.79	34.0	1453.82	0.0	1155.13
13	1485.67	0.0	2.72	1482.02	0.0	7.91	1624.16	35.3	1481.40	0.0	1287.18
14	1611.99	0.0	2.67	1658.55	0.0	7.42	1760.51	37.1	1624.96	0.0	1353.08
15	1748.31	0.0	2.66	1892.89	0.0	7.42	2105.02	33.6	1858.87	0.0	1457.61
16	1055.23	0.0	1.98	1383.57	0.0	3.53	1288.48	10.3	1267.87	0.0	1002.49
17	1117.22	0.0	1.64	1416.14	0.0	3.31	1314.09	9.4	1261.17	0.0	1042.35
18	1216.24	0.0	1.77	1614.11	0.0	2.91	1383.19	10.8	1366.21	0.0	1129.16
19	874.04	0.0	0.86	919.59	0.0	1.34	1146.74	22.0	969.96	0.0	813.50
20	950.72	0.0	0.78	972.76	0.0	1.24	1144.96	24.0	1140.47	0.0	848.93
21	1009.38	0.0	0.75	1096.08	0.0	1.31	1263.70	57.0	1174.43	0.0	909.06
Avg	1105.09	0.0	1.26	1171.38	5.24	2.68	1226.90	25.93	1166.86	0.0	959.14
ARPD	15.22	—	—	22.13	—	—	27.92	—	21.66	—	0.00

[a]Best solutions from 46 runs.
[b]Best solutions from n runs.
[c]Average solutions from 10 runs.
[d]Total times in seconds on a Pentium IV 1.5 GHz PC.
[e]Runtimes not available.

Even with the maximum number of possible restarts n, the results of T-Sweep were still slightly worse in comparison to the T-Cluster heuristic. It failed in designing feasible solutions for all test problems and has the drawback that the number of multi-starts and thereby the runtime increases with the number of customers.

It should be mentioned that the T-Cluster heuristic created very good solutions to the clustered problems 16–21, indicating that the distance to the next route member in the selection strategy is a valuable component. Based on these results, the T-Cluster heuristic will be used to construct a starting solution in the proposed tabu search approach.

5.2. Results of the tabu search heuristic

5.2.1. Sensitivity analysis

The tuning of the tabu search was performed sequentially for every parameter, leaving the remaining parameters unchanged. Tuning began by finding the best move types. Then, the other parameters were tuned in the following order (with their initial value in brackets): neighborhood size h $(h=15)$, sampling size ω $(\omega=1)$, long-term penalty adjuster γ $(\gamma=0)$, shifting penalty adjuster δ $(\delta=0.5)$, tabu duration multiplier $\bar{\theta}$ ($\bar{\theta}$=8), intensification strategy I_1 and I_2. For all, except tuning of the intensification strategies, no restarts were performed, that is $\mu=\infty$. Furthermore, no descent search was applied during sensitivity analyses.

Because of the huge number of possible test runs, the algorithm was tested on a representative subset of 9 out of the 21 benchmark problems, selecting problems 4–6, 10–12 and 19–21. For each of the selected 9 benchmark problems in total about 280 runs were performed during sensitivity analysis (this number includes that a parameter setting was run 10 times on the same benchmark problem whenever a random component ($\omega \neq 1$) was applied). A time limit of 20 min was set for each run instead of a limit on the number of iterations.

Testing the definition of the neighborhood, the tests were restricted to shift and swap subsets with a maximum number of consecutive nodes $v_1, v_2 \leqslant 2$. Results showed that a combination of shift and swap moves clearly outperforms pure shift moves and that the root refining move is a valuable component. Furthermore, the neighborhoods were $v_1=2$ always showed the best performance. In addition a variable neighborhood definition was studied by using a *basic neighborhood* $N_1(s)$ and, whenever the current solution s is close to the best solution s^*, a larger *intensification neighborhood* $N_2(s)$, to more thoroughly examine the search space close to s. The criterion for setting the neighborhood type was defined as follows: if $c(s) > (1+0.1/\sqrt{n}) \cdot c(s^*)$, $N_1(s)$ is used; otherwise $N_2(s)$. Note that n denotes the number of customers in the problem at hand and that $1/\sqrt{n}$ normalizes the selection criterion to the problem size, with a tendency to $N_1(s)$ in greater problems and $N_2(s)$ in smaller problems, as for example desired for CVR improvement. The best results were obtained by using in both $N_1(s)$ and $N_2(s)$ a value of $v_1=2$ as well as the root refining move and to switch from $v_2=1$ in $N_1(s)$ to $v_2=2$ in $N_2(s)$. The variable neighborhood definition clearly outperformed the static neighborhoods, a result also experienced by Osman and Wassan [25,26]. They used the control mechanism of reactive tabu search to switch between an intensification and diversification neighborhood. In fact, using a larger neighborhood sparingly helps to significantly cut down CPU times. The variable neighborhood was fixed for this heuristic.

Next, tests were ran on the neighborhood size parameter h for values of 5,8,10,15,20 and 25. The best results for this critical parameter were found where $h=15$ and the solution quality was found worse the larger the deviation from this value.

Having set $h=15$, the size of the candidate set was tuned, determined by parameter ω. The values of 2–4 were compared with the intial setting without sampling $\omega=1$. A value of $\omega=4$ seemed to be too restrictive. However, no clear statement could be drawn on the other parameter values. Therefore the efficiency of a long-term memory as a diversification generator was examined both for use with ($\omega \in \{2,3\}$) and without sampling ($\omega=1$). Note that the long-term memory was not applied up to this point, as a value of $\gamma=0$ was set initially. For each of the three ω values, parameter γ was examined for the values of 10^{-1}, 10^{-2}, 10^{-3} and 10^{-4}. The best parameter setting was found to be $\gamma=10^{-3}$ and $\omega=3$. This setting was fixed for further tuning runs.

Then, δ was studied for values of 0.1, 0.5, 1.0 and 2.0. The most appropriate value was found to equal the initial setting of $\delta=0.5$, which was set based on the results of Cordeau et al. [22] for their algorithm. Please recall that the shifting penalty parameter α was limited to be between [0.01; 100].

Based on preliminary studies, the tabu duration was set as $\theta=\min\{[0.3n]; [\bar{\theta} \cdot \log_{10}(n)]\}$, where n denotes the number of customers in the problem at hand. Using the factor $\log_{10}(n)$ reflects that the tabu duration should increase slowly with the size of the problem. To avoid the tabu duration from becoming too restrictive when n is very small, as may be the case when applied to a CVR only, a limit of $0.3n$ was included in the definition. The parameter $\bar{\theta}$, initially set to $\bar{\theta}=8$, was then tested for whole numbers from 5 to 10 and random values in the range from 6 to 8. The search did not react very sensitive to the value of $\bar{\theta}$, possibly due to the sampling approach that helps avoid cycling. Parameter $\bar{\theta}$ was fixed to a constant value of 6.

All tunings up to this point were done with strategy I_0. Next the restart strategies I_1 and I_2 were examined for values of $\mu = 2n$ and $\mu = 5n$. Whereas the I_1 strategy performs quite similar to I_0, the advanced intensification strategy I_2 is able to improve upon both of the other strategies. Best results were obtained for I_2 where $\mu=5n$. This indicates that a good structure for all CVRs can hardly be found during a normal search applied to the entire problem and that the intermediate CVR improvement is an essential component.

5.2.2. Comparison of the tabu search algorithm

The tabu search heuristic was run on all 21 test problems for the TTRP with the best parameter settings determined during sensitivity analysis: $N_1(s) = (v_1 = 2;\ v_2 = 1;$ root refining$)$, $N_2(s) = (v_1 = 2;\ v_2 = 2;$ root refining$)$, $h = 15$, $\omega = 3$, $\gamma = 0.001$, $\delta = 0.5$, $\bar{\theta} = 6$ and the I_2 strategy with $\mu = 5n$. Instead of a time limit, a limit on the maximum number of iterations of $\eta = 15\,000$ was set. This was found to be a good compromise between solution quality and runtime of the algorithm.

For each of the 21 test problems, results obtained by the new tabu search method, by the method of Chao [4], as well as the cost $c(s^{**})$ of the best known solution s^{**} found by the new algorithm during sensitivity analysis are reported in Table 4. The best solution found by the T-Cluster heuristic was used as the initial solution for the tabu search heuristic. The T-Cluster values are given in the first main column. All solutions are already feasible and all VC customers are located in main-tours. Because of the sampling strategy, the tabu search was run 10 times on each test instance. The second and third main column show the details for the new tabu search at iterations $\lambda = 1000$ and $\lambda = 15\,000$, respectively. For each test problem, the columns provide the cost of the best solution found during the 10 runs (min $c(s^*)$), the average objective function value over the best solutions from the 10 runs (avg $c(s^*)$) and the average time (T) from 10 runs of the algorithm (including the total time to construct the starting solution with the T-Cluster heuristic). Main column four refers to the tabu search method of Chao. For each test problem it provides the objective function value of the best solution found during 10 runs of Chao's tabu search (min $c(s^*)$) with his best parameter set 5 (see Chao [4, p. 48]), the time (T) for set 5 (note that it is unclear if the time given refers to the total time for all 10 runs or to the time for a representative single run) and the best solution value found by Chao over all his parameter combinations (best $c(s^*)$). Note also that the solution reported by Chao for instance 15 is infeasible, because of an incorrect CVR structure. The last two rows present the average (avg) and the average relative percentage deviation (ARPD) for all 21 problems, where applicable.

Care must be taken when comparing these two algorithms, because different starting solutions were used. The proposed tabu search always uses the best solution found from 46 runs of the T-Cluster heuristic as the starting solution. The method of Chao creates 10 different initial solutions. However, it remains unclear whether Chao's tabu search starts 10 times from the best initial solution found during 10 initial construction runs or if the tabu search algorithm is started once on each of the 10 different initial solutions. Only the average objective function values from the 10 initial solutions are available to us (see Table 3). Furthermore, a direct comparison of runtimes is not possible, since a different computer was used. According to Dongarra [27] our computer (Pentium IV 1.5 GHz PC) has a speed factor of

Table 4
Results of the tabu search heuristic

ID	T-Cluster		$\lambda = 1000$			$\lambda = 15\,000$			Chao			$c(s^{**})$
	$c(s^*)$	T[a]	Min $c(s^*)$[b]	Avg $c(s^*)$[c]	T[d]	Min $c(s^*)$[b]	Avg $c(s^*)$[c]	T[d]	Min $c(s^*)$[e]	T[f]	Best $c(s^*)$[g]	
1	651.87	0.01	567.29	582.62	0.62	566.80	567.98	9.51	565.02	4.19	565.02	564.68
2	697.51	0.01	618.47	644.26	0.57	615.66	619.35	9.60	662.84	5.22	658.07	612.75
3	766.25	0.01	641.26	669.11	0.73	620.78	629.59	11.24	664.73	6.50	648.74	618.04
4	979.79	0.01	811.68	820.53	1.31	801.60	809.13	18.49	857.84	7.53	856.20	798.53
5	1037.50	0.01	860.10	890.60	1.14	839.62	858.98	15.16	949.98	7.06	949.98	839.62
6	1173.11	0.01	956.10	984.48	1.28	936.01	949.89	18.62	1084.82	7.96	1053.23	933.26
7	904.77	0.02	840.27	844.43	2.75	830.48	832.91	33.60	837.80	16.43	832.56	830.48
8	965.90	0.02	882.16	892.35	2.05	878.87	881.26	25.66	906.16	11.11	900.54	878.36
9	1081.21	0.02	959.42	970.08	2.68	942.31	955.95	30.47	1000.27	10.18	971.62	934.47
10	1167.38	0.03	1057.69	1076.58	5.54	1039.23	1052.65	60.94	1076.88	21.72	1073.50	1039.07
11	1274.67	0.03	1118.78	1138.13	5.66	1098.84	1107.47	56.17	1170.17	17.10	1170.17	1094.11
12	1438.11	0.03	1187.53	1204.22	6.33	1175.23	1184.58	63.71	1217.01	20.27	1217.01	1155.13
13	1485.67	0.05	1325.38	1332.94	16.80	1288.46	1296.33	165.41	1364.50	42.34	1364.50	1287.18
14	1611.99	0.04	1400.60	1419.32	14.48	1371.42	1384.13	132.06	1464.20	25.96	1464.20	1353.08
15	1748.31	0.04	1533.47	1540.85	16.98	1459.55	1488.71	154.10	1544.21	24.62	*1540.25*[h]	1457.61
16	1055.23	0.03	1003.69	1008.09	3.06	1002.49	1003.00	43.14	1064.89	14.56	1041.36	1002.49
17	1117.22	0.03	1046.40	1050.89	2.53	1042.35	1042.79	33.73	1104.67	13.74	1090.46	1042.35
18	1216.24	0.03	1147.40	1150.39	2.96	1129.16	1141.94	31.78	1202.00	12.52	1141.36	1129.16
19	874.04	0.01	814.60	819.49	1.99	813.50	813.98	28.84	887.22	16.13	854.02	813.50
20	950.72	0.01	852.27	861.75	1.83	848.93	852.89	24.57	963.06	10.09	942.39	848.93
21	1009.38	0.01	913.19	933.39	2.22	909.06	914.04	26.84	952.29	9.57	926.47	909.06
Avg	1105.09	0.02	977.99	992.12	4.45	962.40	970.84	47.32	1025.74	14.51	1012.46	959.14
ARPD	15.22	—	1.97	3.44	—	0.34	1.22	—	6.95	—	5.56	0.00

[a]Total times for 46 runs in minutes on a Pentium IV 1.5 GHz PC.
[b]Best solutions from 10 runs.
[c]Average solutions from 10 runs.
[d]Average times from 10 runs in minutes on a Pentium IV 1.5 GHz PC.
[e]Best solutions from 10 runs for Chao's set 5.
[f]Times in minutes on a Pentium II 350 MHz PC for Chao's set 5.
[g]Best solutions found by Chao.
[h]Reported solution is infeasible.

around 326 Mflop/s (millions of floating-point operations per seconds), while Chao's computer (Pentium II 350 MHz PC) has a speed factor of around 70 Mflop/s (compared to an Intel Pentium II 333 MHz computer with 69 Mflop/s), that is our computer may be considered around 5 times faster than Chao's computer. Regarding the times given, it seems that the new heuristic is very time consuming. However, it should be mentioned that the algorithm is embedded into a solution-framework that allows for additional side constraints, as for example additional cost components, multiple depots, and a planning horizon of several days. It is assumed that a stand-alone implementation of the algorithm would lead to significantly reduced runtimes.

Keeping the above-mentioned aspects in mind, the new algorithm seems to perform competitive to the tabu search algorithm of Chao. For example, a comparison of Chao's best results from 10 runs for his best parameter set 5 with the best solutions from 10 runs of the new heuristic shows, that the new algorithm was able to obtain better solutions for 19 out of the 21 problems already after 1000 iterations. Furthermore, the proposed tabu search heuristic found new best solutions to each of the 21 test problems during sensitivity analysis. Details of the new best solutions can be found in Scheuerer [28]. The costs of the new best solutions are reported in the last column of Table 4.

6. Conclusion

Two new construction heuristics were proposed along with a tabu search algorithm for the truck and trailer routing problem. Details on sensitivity analyses for the tabu search parameters were presented and a comparison to existing approaches was performed based on instances from the literature. The new construction heuristics proved to successfully find good initial solutions for this variant of the VRP. Computational results also indicate that the new tabu search algorithm is competitive to previously reported heuristics. It obtained better solutions for each of the 21 test problems.

Acknowledgements

The author would like to thank the referees for their valuable comments that helped to significantly improve this paper.

References

[1] Toth P, Vigo D, editors. The vehicle routing problem. Philadelphia, PA: SIAM Monographs on Discrete Mathematics and Applications; 2002.
[2] Laporte G, Gendreau M, Potvin J-Y, Semet F. Classical and modern heuristics for the vehicle routing problem. International Transactions in Operational Research 2000;7:285–300.
[3] Cordeau J-F, Gendreau M, Laporte G, Potvin J-Y, Semet F. A guide to vehicle routing heuristics. Journal of the Operational Research Society 2002;53:512–22.
[4] Chao I-M. A tabu search method for the truck and trailer routing problem. Computers and Operations Research 2002;29: 33–51.
[5] Semet F, Taillard E. Solving real-life vehicle routing problems efficiently using tabu search. Annals of Operations Research 1993;41:469–88.
[6] Semet F. A two-phase algorithm for the partial accessibility constrained vehicle routing problem. Annals of Operations Research 1995;61:45–65.

[7] Fisher ML, Jaikumar R. A generalized assignment heuristic for vehicle routing. Networks 1981;11:109–24.
[8] Gerdessen JC. Vehicle routing problem with trailers. European Journal of Operational Research 1996;93:135–47.
[9] Laporte G, Semet F. Classical heuristics for the capacitated VRP. In: Toth P, Vigo D, editors. The vehicle routing problem. Philadelphia, PA: SIAM Monographs on Discrete Mathematics and Applications; 2002. p. 109–28.
[10] Van Breedam A. A parametric analysis of heuristics for the vehicle routing problem with side-constraints. European Journal of Operational Research 2002;137:348–70.
[11] Lin S. Computer solutions to the traveling salesman problem. Bell System Technical Journal 1965;44:2245–69.
[12] Or I. Traveling salesman-type combinatorial optimization problems and their relation to the logistics of regional blood banking. Dissertation, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL, 1976.
[13] Gillett BE, Miller LR. A heuristic algorithm for the vehicle dispatch problem. Operations Research 1974;22:340–9.
[14] Glover F. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research 1986;13:533–49.
[15] Glover F, Laguna M. Tabu search. In: Reeves CR, editor. Modern heuristic techniques for combinatorial problems. London, Edinburgh, Boston, Melbourne, Paris, Berlin, Wien: Blackwell; 1993. p. 70–150.
[16] Glover F, Laguna M. Tabu search. Boston, Dordrecht, London: Kluwer Academic Publishers; 1997.
[17] Gendreau M. An introduction to tabu search. In: Glover F, Kochenberger GA, editors. Handbook of metaheuristics. Boston, Dordrecht, London: Kluwer Academic Publishers; 2003. p. 37–54.
[18] Cordeau J-F, Laporte G. Tabu search heuristics for the vehicle routing problem. Technical Report G-2002-15, Group for Research in Decision Analysis (GERAD), Montreal, 2002.
[19] Glover F, Kochenberger GA, editors. Handbook of metaheuristics. Boston, Dordrecht, London: Kluwer Academic Publishers; 2003.
[20] Osman IH, Kelly JP. Meta-Heuristics: an overview. In: Osman IH, Kelly JP, editors. Meta-Heuristics: Theory and Applications. Boston, Dordrecht, London: Kluwer Academic Publishers; 1996. p. 1–21.
[21] Gendreau M, Hertz A, Laporte G. A tabu search heuristic for the vehicle routing problem. Management Science 1994;40:1276–90.
[22] Cordeau J-F, Gendreau M, Laporte G. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 1997;30:105–19.
[23] Osman IH. Meta strategy simulated annealing and tabu search algorithms for the vehicle routing problem. Annals of Operations Research 1993;41:421–51.
[24] Taillard E. Parallel iterative search methods for vehicle routing problems. Networks 1993;23:661–73.
[25] Osman IH, Wassan NA. A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. Journal of Scheduling 2002;5:263–85.
[26] Wassan NA, Osman IH. Tabu search variants for the mix fleet vehicle routing problem. Journal of the Operational Research Society 2002;53:768–82.
[27] Dongarra JJ. Performance of various computers using standard linear equations software. Technical Report CS-89-85, Department of Computer Science, University of Tennessee, Knoxville, TN, April 9, 2004. Internet: http://www.netlib.org/benchmark/performance.ps.
[28] Scheuerer S. Neue Tabusuche-Heuristiken für die logistische Tourenplanung bei restringierendem Anhängereinsatz, mehreren Depots und Planungsperioden. Dissertation, School of Business, University of Regensburg, Germany, 2004 [in German].