Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Computers & Operations Research 32 (2005) 1379–1407
computers & operations research
www.elsevier.com/locate/dsw

A TABU search heuristic for the team orienteering problem

Hao Tang[a], Elise Miller-Hooks[b,*]

[a] *Operations Research and Spatial Applications, FedEx Express Corporation, 3680 Hacks Cross Road, Memphis, TN 38125, USA*
[b] *Department of Civil and Environmental Engineering, The University of Maryland, 1173 Glenn L. Martin Hall, College Park, MD 20742, USA*

Abstract

This paper describes a tabu search heuristic for the Team Orienteering Problem (TOP). The TOP is a variant of the well-known Vehicle Routing Problem in which a set of vehicle tours are constructed such that the total collected reward received from visiting a subset of customers is maximized and the length of each vehicle tour is restricted by a pre-specified limit. The tabu search heuristic is embedded in an adaptive memory procedure that alternates between small and large neighborhood stages during the solution improvement phase. Both random and greedy procedures for neighborhood solution generation are employed and infeasible, as well as feasible, solutions are explored in the process. Results from computational experiments conducted on a set of published test problems show that the proposed technique consistently produces high-quality solutions and outperforms other published heuristics for the TOP.

Keywords: Team Orienteering Problem (TOP); Multiple Tour Maximum Collection Problem; Tabu Search; Adaptive Memory Procedure; Selective TSP

1. Introduction

In this paper, a tabu search heuristic is proposed to solve the Team Orienteering Problem (TOP). Given a complete graph $G=(V,E)$, where $V=\{0,1,2,\ldots,n-1\}$ is the vertex set, $E=\{(i,j)\,|\,i,j\in V\}$ is the edge set and r_i is the reward collected by visiting vertex i ($i\in V\setminus\{0\}$), the TOP seeks a set of m vehicle tours that start at the same vertex, collect the maximum total reward by visiting select vertices within a pre-specified duration limit L, and return to the ending vertex. For the purpose of simplifying the notation, we assume that these m tours have an identical starting and terminus point, vertex 0. No capacity constraints are considered.

* Corresponding author. Tel.: 301-405-2046; fax: 301-405-2585.
E-mail address: elisemh@umd.edu (E. Miller-Hooks).

0305-0548/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2003.11.008

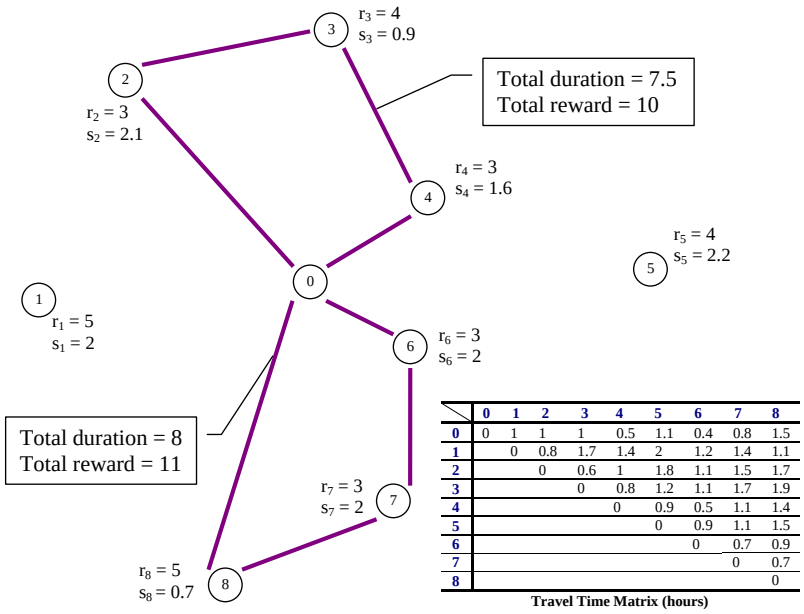

Fig. 1. An illustrative example.

The TOP arises in many applications. Consider, for instance, routing technicians to service customers at geographically distributed locations. In this context, each vehicle in the TOP model represents a single technician and there is often a limitation on the number of hours each technician can be scheduled to work in any given day. Thus, it may not be possible to include all customers requiring service in the technicians' schedules for a given day. Instead, a subset of the customers will be selected. Decisions regarding which customers to choose for inclusion in each of the service technicians' schedules may take into account customer importance or task urgency. Note that this customer selection requirement also arises in many real-time routing applications.

The service scheduling application of the TOP is illustrated for two technicians (i.e. $m = 2$) in Fig. 1. In the figure, vertices 1–8 represent customer requests for service. The relative location of each vertex corresponds with the geographical location of each request. Suppose that the maximum working duration (travel time plus service time) for each day is 8 h (i.e. $L = 8$ h). For the purpose of simplification, breaks are ignored. Service time s_i and reward r_i for each vertex i ($i \in [1, \ldots, 8]$) and travel times between requests are shown in the figure. Travel times are assumed to be symmetric. Rewards are greater for servicing requests with greater urgency or higher priority. The problem is to select a set of requests to be serviced on this day by each of the two technicians and to determine the order in which they will be served such that the overall reward is maximized. Each technician must be able to service all assigned requests within the tour duration limit L. Note that there may be some savings incurred by servicing a less urgent request, and thus, it is not necessarily best to only consider the requests with the highest rewards (i.e. with greatest urgency or priority).

The optimal solution for this illustrative problem instance is depicted in the figure. The solution contains two tours resulting in 21-unit total reward. Note that two requests (located at vertices

1 and 5) are not included in either technician's schedule. Due to the tour duration restrictions, including these requests would require exclusion of other requests and would result in a lower total reward.

Problems arise in many other applications that can be similarly modeled as TOPs, including the multi-vehicle version of the home fuel delivery problem [1], the sport game of team orienteering [2] and athlete recruiting from high schools for a college team [3]. In addition, some applications of pickup or delivery services involving the use of common carriers and private fleets also require selection of only a subset of available customers (see, e.g. [4–6]).

The TOP has received some attention in the published research works. In these works, the TOP is also referred to as the Multiple Tour Maximum Collection Problem (MTMCP), where all tours have an identical starting and terminus vertex [3]. Two heuristics have been previously proposed for the TOP: a greedy construction procedure [3] and the 5-step heuristic [2]. In the former procedure, at each iteration the best pair of vertices are assigned to the solution tours. The procedure stops when all m tours have been constructed and it does not appear possible to add additional vertices without exceeding tour length restrictions. The latter 5-step (initialization, main movement, clean up, local improvement and reinitialization) heuristic can be viewed as a deterministic variant of simulated annealing [7], a type of metaheuristic. The 5-step metaheuristic proposed by Chao et al. appears to be the best published heuristic available for the TOP. The only exact algorithm that addresses the TOP is based on column generation [8]. This procedure is capable of solving some small- to moderate-size problems with up to 100 vertices, provided the number of vertices in each tour remains relatively small.

The TOP is the multi-vehicle version of the Selective Traveling Salesperson Problem (STSP). A number of works have addressed the STSP or the related Orienteering Problem (OP, the single vehicle version of the TOP), several of which propose exact algorithms based on branch-and-bound [9,10] and branch-and-cut [11,12] procedures. The most successful implementation was due to the branch-and-cut procedure (e.g., [11]), which solved STSP instances with up to 500 vertices.

As shown by Golden et al. [1], the OP (and thus, the STSP) is NP-hard. Therefore, most research on these problems have focused on providing heuristic approaches. Tsiligirides [13] proposed deterministic and stochastic heuristics for the problem. The deterministic heuristic uses a variant of the heuristic procedure proposed by Wren and Holliday [14] for vehicle routing and the stochastic method relies on Monte Carlo techniques. A center-of-gravity heuristic was introduced by Golden et al. [1], where the solution tour is constructed by the cheapest insertion procedure according to a combined measure for vertex selection. Incorporating the concept of a learning measure, Golden et al. [15] improved the center-of-gravity heuristic by rewarding vertices associated with "above-average" tours while penalizing those associated with "below-average" tours. Ramesh et al. [16] proposed a four-phase heuristic. After choosing the best solution from iterations over a set of three phases (vertex insertion, edge exchange and vertex deletion), a fourth phase is entered, where one attempts to insert unvisited vertices into the tour. A 5-step heuristic was proposed by Chao et al. [17] for the OP. This heuristic takes advantage of a probabilistic criterion for candidate solution selection. Gendreau et al. [12] developed two approximate algorithms for the STSP. The first, H1, gradually constructs a solution tour by inserting a single or a pair of vertices into the current tour. The second, H2, constructs a tour using all vertices, then gradually removes some vertices from the tour. A tabu search procedure that incorporates the proximity measure on clusters of vertices in

candidate moves, rather than on single vertices, was proposed by Gendreau et al. [18]. Numerical experiments show that this tabu search heuristic is able to provide close to optimal solutions for the STSP.

The prize-collecting traveling salesperson problem is another related problem to the TOP that employs the use of rewards or profits for visiting customers (e.g. [19]). In this problem class, a minimum-cost route is constructed such that the route visits each customer at most once and the total collected prize (or profit) is no less than a given value.

Compared to the closely related STSP and Vehicle Routing Problem (VRP), the TOP is a much less explored problem, despite that there are a wealth of applications that could benefit from such a model. Unfortunately, techniques developed for these related problems cannot be directly applied or easily extended to solve the TOP or its variants.

In the STSP, a subset of customers is chosen and the optimal order in which to visit these customers is simultaneously determined. In contrast to the STSP, in the TOP, in addition to selecting the subset of customers to be visited and determining the optimal order in which to visit them, these customers must also be assigned to the m vehicles. Thus, the joint selection and ordering problem for a single tour becomes a joint selection, assignment and ordering problem for m tours. The increased complexity in problem structure of this multiple vehicle version of the STSP that results from the additional consideration of the assignment of customers to multiple tours significantly amplifies the search space for exactly or approximately solving this routing problem (the TOP) as compared with its single vehicle version (the STSP).

It is well known that the m-TSP is no more difficult than the TSP, because an m-TSP instance can be easily transformed to a TSP instance (see, e.g. [20]). Unfortunately, a similar relationship between the TOP (or MTMCP) and the STSP is not known (the method for transforming an m-TSP to a TSP is not valid for creating a similar transformation between the TOP and the STSP due to the existence of tour duration limitations).

The TOP also differs from the VRP in a significant way. Thus, existing procedures for addressing the VRP cannot be directly applied to solve the TOP. In the VRP, all customers must be included in the final set of tours; whereas, in the TOP, the final set of tours will contain only a subset of the customers as a consequence of tour duration limitations. Furthermore, in the VRP, the objective is to find the set of tours that minimize the total travel cost; whereas, in the TOP the objective is to maximize the total reward received by visiting the chosen subset of customers.

Due to the intractability of the STSP and VRP, most research efforts that address these problems focus on the development of heuristics that solve them approximately, but efficiently (as compared to exact approaches). Among these approximate approaches, metaheuristics have gained recent popularity due to their ability to provide close to optimal solutions within reasonable computing time (e.g. [21,22]). Although tabu search has been found to be one of the most promising metaheuristics for the STSP and the VRP, it appears that no tabu search algorithms have been previously proposed in the literature for the TOP.

In this paper, an integer programming formulation for the TOP is provided (Section 2) and a tabu search heuristic is devised. A step-by-step description of this procedure is provided (Section 3). Computational results on published test problems are presented (Section 4). The results of these computational experiments show that the proposed tabu search algorithm for solving the TOP outperforms other published heuristics for the TOP. Conclusions follow (Section 5).

2. Mathematical formulation

The TOP described in Section 1 can be formulated as an integer program using the following additional notation.

y_{ik}	1, if vertex i is visited by vehicle k; 0, otherwise
x_{ijk}	the number of times edge (i,j) is traversed by vehicle k; $x_{ijk} \in \{0,1\}$ if $i,j \in V \setminus \{0\}$, $x_{ijk} \in \{0,1,2\}$ if $i=0$ and $j \in V \setminus \{0\}$.
d_{ij}	distance of travel along edge $(i,j), (i,j) \in E$
s_i	service time at vertex i, $i \in V$
U	a subset of vertices in V, i.e. $U \subset V$

Note that $x_{0jk} = 2$ is defined for a tour that traverses only one customer vertex in addition to depot 0, i.e. the tour consists of two edges: $(0,j)$ and $(j,0)$. In this formulation, the distance matrix is assumed to be symmetric, i.e. $d_{ij} = d_{ji}$ for all $(i,j) \in E$; thus, only x_{ijk} $(i<j)$ and d_{ij} $(i<j)$ are defined. The formulation of the TOP is given next (for $n > 3$):

$$\text{Max} \quad \sum_{i=1}^{n-1} \sum_{k=1}^{m} r_i y_{ik} \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n-1} \sum_{k=1}^{m} x_{0jk} = 2m, \tag{2}$$

$$\sum_{i<j} x_{ijk} + \sum_{i>j} x_{jik} = 2y_{jk} \ (j = 1,2,\ldots,n-1; k = 1,2,\ldots,m), \tag{3}$$

$$\sum_{i=0}^{n-2} \sum_{j>i} d_{ij} x_{ijk} + \sum_{i=1}^{n-1} s_i y_{ik} \leqslant L \ (k = 1,2,\ldots,m), \tag{4}$$

$$\sum_{k=1}^{m} y_{ik} \leqslant 1 \ (i = 1,2,\ldots,n-1), \tag{5}$$

$$\sum_{\substack{i,j \in U \\ i<j}} x_{ijk} \leqslant |U| - 1 (U \subset V \setminus \{0\}; n-2 \geqslant |U| \geqslant 2; k = 1,2,\ldots,m), \tag{6}$$

$$x_{0jk} \in \{0,1,2\} (j = 1,2,\ldots,n-1; k = 1,2,\ldots,m), \tag{7}$$

$$x_{ijk} \in \{0,1\} (1 \leqslant i < j \leqslant n-1; k = 1,2,\ldots,m), \tag{8}$$

$$y_{ik} \in \{0,1\} (i = 1,2,\ldots,n-1; k = 1,2,\ldots,m). \tag{9}$$

The objective of the TOP is to maximize the total reward of all vehicle tours, as shown in (1). In this formulation, constraint (2) ensures that there are m tours starting and ending at depot 0. Constraints (3) guarantee the connectivity of each vehicle tour, while the limitation on total duration for each tour is respected by constraints (4) (note that constraint (4) may also be written

as $\sum_{i=0}^{n-2}\sum_{j>i}(d_{ij}+\frac{s_i}{2}+\frac{s_j}{2})x_{ijk} \leqslant L$ $(k=1,2,\ldots,m)$). Constraints (5) ensure that all vertices (except for depot 0) can be visited at most once. Sub-tours are prohibited by (6). Constraints (7)–(9) set integral requirements on the decision variables. Another integer formulation for the TOP has been presented in the literature [3] with $2n^2m$ integer variables. The formulation presented in this paper has only $(n^2m+(n-2)m)/2$ integer variables.

The TOP formulated in (1)–(9) is NP-hard. This is because when $m=1$, the TOP reduces to an STSP, which is NP-hard [1].

3. The TABU search algorithm

In this section, a tabu search heuristic for solving the TOP is described. Introduced and refined by Glover [23–25], tabu search is a type of metaheuristic that has been widely used to solve complex combinatorial optimization problems. As with many other metaheuristics, the success of tabu search is in large part due to its ability to steer the search process from getting stuck in a local optimum. In tabu search, this is achieved by allowing a move to a neighboring solution that may result in deterioration in the objective value, but that simultaneously avoids cycling back through previous moves. Specifically, tabu search procedures exploit the short-term memory, i.e. the tabu list, which keeps track of recently visited solutions or their attributes. A move to a neighboring solution is permitted if the neighboring solution is neither contained in the tabu list nor possesses an identical attribute (e.g. objective value) to a solution in that list. However, a move to a neighboring solution could be selected based on some aspiration criterion even if it is prohibited by the tabu list. For example, in most tabu search applications, a particular move may be permitted even if it (or its attribute) is contained in the tabu list as long as such a move will result in a solution that is superior to the best solution obtained thus far.

3.1. Overview of the approach

The tabu search heuristic proposed herein for the TOP can be roughly characterized into three steps: initialization, solution improvement and evaluation. An overview of the approach is depicted in Fig. 2. The tabu search procedure is called from within an Adaptive Memory Procedure (AMP). Details of the AMP are given in Section 3.2.2.

3.2. Step-by-step description

In this section, details of the AMP and tabu search procedure are provided, including details of the initialization, improvement and evaluation (*Steps A*, *B* and *C* in Fig. 2) steps.

3.2.1. Parameters

The tabu search procedure employs a set of parameters whose values need to be set before the algorithm is run. These parameters include the number of tabu iterations (α), penalty factor (η), neighborhood size (β), tour improvement frequency (χ), retrospect length (δ), tour duration violation coefficient (ε), tour selection parameter ($\mathbf{T}$), the number of randomized improvement iterations (κ),

Step A. (Initialization from the AMP) Given current solution S determined in the AMP, set tabu parameters to a small neighborhood stage in which only a small number of neighborhood solutions will be explored.

Step B. (Improvement) Generate by both random and greedy procedures a number of neighborhood solutions (feasible and infeasible) to the current solution based on the current tabu parameters. Note that in select iterations, the sequence of each of these neighborhood solutions is improved by heuristic procedures.

Step C. (Evaluation) Select the best non-tabu solution from the candidates generated in *Step B* (tabu status can be overridden if the best tabu solution is better than the current best feasible solution). Depending on the current neighborhood size and solution quality, set the neighborhood size parameter to large or small stages and return to *Step A* or *B*.

Fig. 2. Tabu search procedure overview.

and tabu tenure interval $[\theta_{\min}, \theta_{\max}]$. The definitions of these parameters are described as follows. Their utility will be explained in more detail in the following subsections.

α: The maximum non-improvement iterations in the tabu search procedure.

η: The penalty coefficient applied to infeasible solutions generated in the tabu search procedure.

β: The number of neighborhood solutions generated in the tabu search procedure.

χ: Solution tour improvement frequency, i.e. these tours will be improved by a post-optimization procedure every χ tabu iterations.

δ: Every δ tabu iterations, the previously selected non-tabu solutions will be examined. If all of them are feasible, penalty coefficient η will be halved; if all of them are infeasible, η will be doubled.

ε: Tour duration limit L is adjusted to $L \times (1 + \varepsilon)$ in more than one procedure employed in the tabu search procedure.

T: The number of candidate tours that will be considered for selection in initial solution construction.

κ: The number of remove and reinsert iterations for the RVI procedure (Section 3.2.3.2).

$[\theta_{\min}, \theta_{\max}]$: Tabu tenure for each recently inserted vertex is randomly chosen from interval $\theta_{\min}$ to $\theta_{\max}$.

3.2.2. Adaptive memory procedure (feeds Step A)

The tabu search procedure is embedded in an AMP. At the beginning, a certain amount of storage space (adaptive memory) is allotted within the AMP. A set of partial solutions (a partial solution is defined herein as a single tour, i.e. one of the required m tours) is generated by heuristic techniques and is stored in the adaptive memory. An initial solution is then constructed by combining select partial solutions, where selection preference is probabilistically biased to those partial solutions with

Step 1. Generate a set of N vehicle tours and store them in the adaptive memory. Each vehicle tour is labeled according to its objective value in terms of total reward. Set the iteration counters: $q_{AMP} = 0$ and $q_{tabu} = 0$.

Step 2. Let candidate list, *CL*, include all N tours in the adaptive memory. Construct a new TOP solution S by combining $\leq m$ vehicle tours in the adaptive memory in an attempt to produce a feasible solution to the TOP. Assign a probability of being selected to each tour based on its objective value. If tour r is selected as one of the m vehicle tours of solution S, remove from *CL* all tours having at least one vertex in common with that of tour r. Repeat this process until $CL = \phi$. If the number of tours in S is equal to m, go to *Step 3*; otherwise, construct (from the prior partial solution) new vehicle tours (until the number of tours in S equals m) using the vertices that are not yet included in S (similar to *Step 2* of Figure 4) and go to *Step 3*.

Step 3. Improve solution S by the tabu search algorithm.

Step 4. Update the solutions maintained in adaptive memory by inserting solution tours of the improved solution S from tabu search determined in *Step 3*. Remove the worst tours (i.e. tours with smallest objective values) from the adaptive memory so that the number of vehicle tours included is still N. Let $q_{AMP} = q_{AMP} + 1$. If $q_{AMP} < \lambda$ (the preset total number of the AMP iterations), go to *Step 2*; otherwise, output the best solution and stop.

Fig. 3. Overview of the AMP.

preferred objective values. The initial solution is improved by the tabu search procedure and the solutions maintained in the adaptive memory will be updated using the components (i.e. vehicle tours) of the improved initial solution. As pointed out by Golden et al. [26], the AMP works similarly to genetic algorithms, with the exception that offspring (in AMP, the new initial solutions) can be generated from more than two parents. The idea of using such an AMP was introduced by Rochat and Taillard [27]. Several published studies show that the AMP is very effective in providing high quality solutions, especially when implemented in conjunction with tabu search (e.g. [27,28]). For recent reviews of the AMP and related methods see Taillard et al. [29] and Glover [30]. The AMP process employed in this research effort is described in Fig. 3, where we see that the tabu search procedure is called in *Step* 3.

In Step 2 of Fig. 3, the algorithm attempts to construct an initial solution by first combining available tours stored in the adaptive memory. Note that this does not guarantee that a complete solution with m tours will be constructed. If the solution built by combining the tours in the adaptive memory has fewer than m tours, new vehicle tours will be constructed using vertices that have not yet been visited in the current partial solution until it contains m vehicle tours and no more vertices can be inserted in the new tours without violating duration limitations. This general framework for initial solution construction was proposed by Rochat and Taillard [27] and successfully implemented to solve several VRP-related problems (e.g. [26,28,31]).

As described in *Step* 2 of Fig. 3, tour selection in initial solution construction is probabilistically biased to those partial solutions with higher objective values. This can be implemented in several different ways. The implementation employed herein is as follows. T partial tours with highest rewards are sorted in a non-increasing order. Each partial tour τ_i $(i = 1, 2, \ldots, T)$ is assigned an index value $Index(i) = \sum_{j=1}^{i} R(\tau_j) / \sum_{j=1}^{T} R(\tau_j)$, where $R(\tau_j)$ is the total reward of tour τ_j. Let

Step 1. Create m vehicle tours $\tau_k = \{0, 0\}$ with durations $D(\tau_k) = 0$ ($k = 1, 2, \ldots, m$). Let $SE = \{j \mid d_{oj} + d_{jo} + s_j \le L, \forall j \in V\}$. Let $k = 1$.

Step 2. If τ_k contains only vertex 0, randomly select $j \in SE$ and insert j in τ_k; otherwise, select $j \in SE$ and two vertices p and q in τ_k such that evaluation function $(d_{pj} + d_{jq} - d_{pq} + s_j)/r_j$ is minimal and the updated duration of vehicle tour τ_k: $D(\tau_k) + d_{pj} + d_{jq} - d_{pq} + s_j \le L$. If no such j can be found, go to *Step 3*; otherwise, insert j between p and q, remove j from SE, update the duration of τ_k, and repeat *Step 2*.

Step 3. Let $k = k + 1$. If $k > m$, stop; otherwise, go to *Step 2*.

Fig. 4. Generating partial solutions for the AMP.

$Index(0) = 0$. During tour selection, a random number between 0 and 1 is generated. If this number is between $Index(i-1)$ and $Index(i)$, then tour τ_i is chosen.

The process of generating a set of N initial vehicle tours in *Step* 1 of the AMP is described in greater detail in Fig. 4. This is a modified cheapest insertion procedure, whose generic form was proposed for the TSP by Rosenkrantz et al. [32]. Specifically, two modifications are made. First, the seed vertex (i.e. the first vertex other than depot 0 to be inserted in each tour) in each tour is randomly selected to diversify produced solutions; second, the evaluation function for candidate vertices is modified such that instead of minimizing total added duration due to inclusion of each additional vertex, we minimize the ratio of added duration to additional reward. To further diversify the solution tours stored in the adaptive memory, a number of alternative solutions may be generated using different evaluation functions. In Fig. 4, it was assumed that evaluation function (10a) was employed:

$$(d_{pj} + d_{jq} - d_{pq} + s_j)/r_j. \tag{10a}$$

Instead, one of the following functions may be employed in place of (10a):

$$(d_{pj} + d_{jq} - d_{pq} + s_j)/\sqrt{r_j}, \tag{10b}$$

$$\sqrt{d_{pj} + d_{jq} - d_{pq} + s_j}/r_j, \tag{10c}$$

$$d_{pj} + d_{jq} - d_{pq} + s_j. \tag{10d}$$

Note that one pass of the procedure defined in Fig. 4 will create and store m feasible vehicle tours in the adaptive memory; thus, executing this procedure $\lceil N/m \rceil$ times will create at least N vehicle tours.

To take advantage of stored partial or complete solutions in the adaptive memory, these solutions must be sorted according to their objective values. A linked list structure is employed to store the initial vehicle tours, which allows for insertion and deletion operations. There are three components for each tour in the linked list: the unit for recording the tour reward, a pointer to the actual address of the stored tour, and the elements of the stored tour. This is shown in Fig. 5.

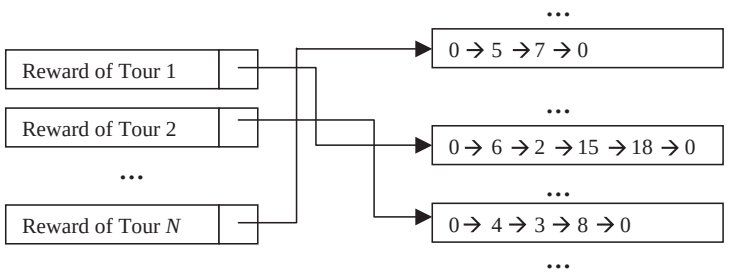

Fig. 5. Data structure for storing feasible solutions in the adaptive memory.

3.2.3. Neighborhood generation and improvement (Step B) in the tabu search

Step B of the tabu search procedure focuses on the generation of neighborhood solutions and improving these solutions. In this process, infeasible solutions (i.e. solutions for which the tour limitation constraint is not enforced) are permitted. Permitting intermediate infeasible solutions may aid in moving the search process out of a local optimum. First introduced by Gendreau et al. [33], empirical results in several different contexts (and also in our numerical experiments) have shown that this strategy is very effective in providing high-quality solutions. In order to guide infeasible solutions to improved feasible solutions through neighborhood moves, a quasi-reward is used in the tabu search heuristic. Here, a quasi-reward of a solution is defined as its true total reward less a penalty cost that is incurred if at least one vehicle route contained in this solution has a duration that is greater than L. The mathematical expression of quasi-reward is presented in expression (11) of Fig. 9. We use a quasi-reward with an adaptively adjusted penalty factor to guide the search process that explores both feasible and infeasible solutions.

In some tabu search implementations (e.g. [34]), the solution improvement phase is divided into two stages: one stage explores a small number of neighborhood solutions (referred to herein as the small neighborhood stage), while the other explores a large number of neighborhood solutions (referred to herein as the large neighborhood stage). The search process enters the large neighborhood stage when no improvement is found in the small neighborhood stage. For example, Barbarosoglu and Ozgur's procedure does not return to the small neighborhood stage until a new initial solution is generated and the tabu search is restarted.

We employ a method of switching between small and large neighborhood exploration stages in our implementation. That is, during the stage where a large number of neighborhood solutions are generated, once an improved solution (over the current best feasible solution) is found, the algorithm immediately returns to the stage where a small number of neighborhood solutions are generated. The algorithm returns to the small neighborhood stage, because an improvement in the best available solution may indicate that the search process has successfully moved out of the current local optimum. Intuitively, one would expect that if the search process stays in the large neighborhood stage (i.e. does not return to small neighborhood stage), the result obtained should be much better than alternating between the two stages. However, empirical results show that alternating between small and large neighborhood stages during the course of the tabu search enables the search to evolve in an efficient way without leading to solutions that degrade in solution quality (see computational experiments (Section 4) for more details). Thus, this tabu search procedure is implemented in such

Step 1. Let $SE = \{j \mid d_{0j} + d_{j0} + s_j \le L, \forall j \in \overline{S}\}$.

Step 2. Select $j \in SE$ and two vertices p and q on τ_k such that evaluation function $d_{pj} + d_{jq} - d_{pq} + s_j$ is minimal. If no such j can be found so that its insertion will maintain the feasibility of tour τ_k, stop. Otherwise, insert j between p and q, remove j from SE, update the duration of vehicle tour $\tau_k : D(\tau_k) + d_{pj} + d_{jq} - d_{pq} + s_j \le L$. Update tabu status of vertex j (see Subsection 3.2.4). Repeat *Step 2*.

Fig. 6. Insertion procedure.

a way that it alternates between small and large neighborhood stages during solution improvement (*Step B* Fig. 2). Note that this alternation in neighborhood solution exploration stages is related to variable depth search (e.g. [35]) and variable neighborhood search [36].

In this step, the algorithm explores various neighborhood moves, such as exchange of vertices that are included in tours with those that are not, exchange of vertices between two different tours, changes in the sequencing of vertices in a tour and addition of vertices not included in tours to existing tours. Both random and greedy procedures are employed in neighbor solution exploration.

3.2.3.1. Neighborhood structure and generating neighborhood solutions. The tabu search procedure starts from an initial solution S constructed in the AMP. In one tabu iteration, neighbors of this initial solution will be generated and improved and the best non-tabu solution from the generated neighborhood solutions will be selected as the initial solution for the next iteration. Similar to most tabu search implementations, a tabu solution will be selected if it is better than the best solution obtained in all previous iterations. Neighbors (or neighborhood solutions) are defined herein as solution tours that have at least one vertex different from the initial solution. If a new solution contains the same set of vertices as the initial solution, but the sequence of these vertices differs between tours, they are not considered neighbors in our implementation.

To generate neighborhood solutions of initial solution S (let $\bar{S}$ be the set of vertices not included in S), we consider inserting additional vertices into S and exchanging vertices between S and $\bar{S}$, as discussed next. Similar neighborhood moves have also been used in other studies, e.g. for the vehicle routing problem with time windows [37].

(1) Insert additional vertices in S. At each iteration, two vehicle tours of S will be randomly selected. Initially, as many vertices of $\bar{S}$ as possible (given the tour duration limitation constraints) are inserted in each of the selected tours. The insertion is conducted via a procedure that is similar to the cheapest insertion for the TSP, as described in Fig. 6 (for tour τ_k).

(2) Exchange vertices between select tours from (1) and $\bar{S}$. For each select tour, randomly choose some vertices and remove them from the tour (the number of vertices to be removed can be chosen randomly, as in our experiments). And for each such tour, randomly insert some vertices from set $\bar{S}$ until the total duration of the select tour is no less than $L(1+\varepsilon)$. Insert these vertices in the location that leads to the minimal increase in tour duration. That is, for a randomly selected $x \in \bar{S}$ and tour τ, x is inserted in τ according to the insertion criterion described in *Step* 2 of Fig. 6.

Step 1. Resequence τ by the largest insertion procedure (Rosenkrantz et al., 1974) with evaluation function $d_{pj} + d_{jq} - d_{pq} + s_j$. Let the resulting tour be τ'. If $D(\tau') < D(\tau)$, let $\tau = \tau'$ and $D(\tau) = D(\tau')$. Set iteration counter $q_{RVI} = 0$ and improvement counter $v = 0$.

Step 2. If $q_{RVI} \geq \kappa$, return τ and stop; otherwise, $q_{RVI} = q_{RVI} + 1$. Randomly remove a subset of vertices from τ' and reinsert them in τ' using the largest insertion criterion. If $D(\tau') < D(\tau)$, let $\tau = \tau'$, $D(\tau) = D(\tau')$ and $v = 0$; otherwise $v = v + 1$. If $v > \kappa/2$, let $\tau' = \tau$. Repeat *Step 2*.

Fig. 7. RVI procedure for tour improvement.

3.2.3.2. Tour improvement. The quality of a feasible TOP solution is solely dependent on the total reward of its constituent vehicle tours (i.e. which vertices have been included in the tours) and not on the duration of each constituent tour. However, improving vertex sequences on solution tours such that the minimum duration is obtained is crucial for achieving high quality solutions. A shorter solution tour is more likely to have more vertices included in successive iterations, and therefore, a higher potential reward at future iterations. Thus, at each iteration, a series of tour improvement procedures that consider tour duration are selectively applied. These procedures include balancing the tours, exchanging vertices between tours, and randomly removing and re-inserting vertices within a tour, as described next.

(1) *Balance between longer and shorter tours*. For randomly selected tours τ_1 and τ_2 from the current solution S, the following improvement attempts are made. If $D(\tau_1) > D(\tau_2)$, scan τ_1 to see if any vertex can be removed from τ_1 and be inserted in τ_2 such that the updated total duration of τ_1 and τ_2 is decreased. If $D(\tau_1) \leqslant D(\tau_2)$, scan τ_2 to see if any vertex can be removed from τ_2 and inserted in τ_1 such that the updated total duration of τ_1 and τ_2 is decreased.

(2) *Exchange of vertices between two tours*. For randomly selected tours τ_1 and τ_2 from the current solution S, this improvement procedure makes one-vertex exchanges between tours τ_1 and τ_2. Starting with the first vertex on τ_1, scan from the first to last vertex on τ_2 to examine if an exchange of the first vertex on τ_1 with the current vertex on τ_2 makes the total duration of the two tours shorter. The exchange is made immediately when such a vertex on τ_2 is found. Then the procedure is repeated with the second vertex on τ_1, scanning vertices on τ_2. This procedure stops when all one-vertex exchanges between τ_1 and τ_2 leading to improved tour duration have been performed.

(3) *A random vertex-insertion (RVI) procedure*. For a select vehicle tour, τ, of duration $D(\tau)$, this semi-greedy random insertion approach randomly removes a subset of vertices from the tour and re-inserts them in the resulting partial tour in a greedy way. This was motivated by the empirical results of Hart and Shogan [38]. Hart and Shogan recognized that there are performance benefits to reverting to a partial solution and reapplying a heuristic. Moreover, they found that additional benefits were achieved by incorporating randomization within the heuristic. A similar remove-and-insert procedure is proposed in Shaw [39] in the context of improving solution routes for the VRP with time windows. A step-by-step description of this approach is shown in Fig. 7, where κ is the preset parameter used to indicate the maximum number of remove-and-reinsert iterations in the RVI. Note that by adjusting the value of κ, the computational time used for solution improvement in the RVI

Step 0. Recall that the current tabu iteration counter is q_{tabu} (initialized in *Step A*). Let neighborhood counter $p = 0$. Set the current initial solution to S and $\overline{S}$ to all vertices not contained in the tours of S.

Step 1. Randomly select two tours τ_1 and τ_2 (associated with two different vehicles) from the current initial solution S.

Step 2. Try to insert additional vertices in tours τ_1 and τ_2 (see procedure (1) of Subsection 3.2.3.1). If any vertices are successfully inserted in τ_1 or τ_2, go to *Step 4*.

Step 3. Exchange vertices between the selected tours (τ_1 and τ_2) and $\overline{S}$ (procedure (2) of Subsection 3.2.3.1).

Step 4. Try to balance between the longer (see procedure (1) of Subsection 3.2.3.2) and shorter tours in τ_1 and τ_2. If the balance is successful, go to *Step 6*.

Step 5. Improve the chosen tours by exchanging vertices between them (procedure (2) of Subsection 3.2.3.2).

Step 6. If iteration counter q_{tabu} is divisible by χ, use the RVI procedure (procedure (3) of Subsection 3.2.3.2) with parameter κ to improve the sequence of each chosen tour. Try to insert additional vertices in each tour (see procedure (1) of Subsection 3.2.3.1) if the tour duration is less than L.

Step 7. Let $p = p + 1$ and $q_{tabu} = q_{tabu} + 1$. If $p < \beta$, go to *Step 1*.

Fig. 8. *Step B* of the tabu search procedure.

can be easily controlled. To reduce computational time, the RVI is applied once every χ iterations to select tours in the tabu search procedure (see Fig. 8 for more detail).

3.2.3.3. Step-by-step description of Step B in the tabu search procedure. Given the description in Sections 3.2.3.1 and 3.2.3.2, a more detailed account of *Step B* in the tabu search procedure can be provided, as given in Fig. 8.

3.2.4. Tour evaluation and selection (Step C) of the tabu search procedure

Neighborhood solutions generated in *Step B* of the tabu search procedure will be evaluated in *Step C*. Specifically, in *Step C*, the best non-tabu solution will be selected as the new initial solution for the next tabu iteration (see Fig. 9). Note that in *Step* 3, the conditions under which the solution improvement procedure in the tabu search heuristic switches between small and large neighborhood stages are presented. These conditions are based on the discussion given in Section 3.2.3.

By allowing neighborhood moves that may result in deterioration in the objective value and by prohibiting cycling to previous moves, tabu search is able to guide the search process from being caught in local optima. More specifically, this is achieved through a tabu list that prevents a move made at iteration q from being repeated until iteration $(q+\theta)$, where θ is the tabu duration that may be set deterministically or chosen randomly on a pre-specified interval. In this study, whenever a vertex is inserted in a solution tour at iteration q, it may not be removed from that tour until iteration $(q+\theta)$, where θ is randomly chosen on interval $[\theta_{\min}, \theta_{\max}]$. Similar to most tabu search procedures

Step 1. For each generated neighborhood solution $\widetilde{S}$, where $\widetilde{S}$ consists of m individual tours τ_k ($k = 1, 2, \ldots, m$), of the initial solution S in *Step B*, calculate its quasi-reward according to the following expression:

$$R^{quasi}(\widetilde{S}) = \sum_{k=1}^{m} R(\tau_k) - \eta \sum_{k=1}^{m} \max(D(\tau_k) - L, 0), \quad (11)$$

where $R(.)$ represents the total reward of a tour. Note that if every tour in $\widetilde{S}$ is feasible (i.e. its duration is no greater than L), then $R^{quasi}(\widetilde{S}) = \sum_{k=1}^{m} R(\tau_k)$.

Step 2. Find the maximum-reward non-tabu and tabu solutions (denoted by S_{nt} and S_t, respectively) out of all neighborhood solutions in *Step B*, based on their quasi-rewards. If the total reward of tabu solution S_t is larger than both the rewards of non-tabu solution S_{nt} and current best solution S^* from all prior iterations of the tabu search procedure, then let $S^* = S_t$, $S = S_t$ and $q_{tabu} = 0$; otherwise, let $S = S_{nt}$. If $S_{nt} > S^*$, set $S^* = S_{nt}$ and $q_{tabu} = 0$.

Step 3. If S is feasible and its total reward is greater than that of the current best feasible solution from all prior iterations of the tabu search procedure, denoted it S_f^*, let $S_f^* = S$ and set $q_{tabu} = 0$. If at least one of the S's found in the last δ iterations are feasible, then set $\eta = \eta/2$; if they are all infeasible, set $\eta = 2\eta$. If the current neighborhood size parameter is small and $q_{tabu} > \alpha/2$, set the neighborhood size parameter to the large size and go to *Step B*; if the current neighborhood size parameter is small and $q_{tabu} \le \alpha/2$, go to *Step B*. If the algorithm is in the large neighborhood stage and $q_{tabu} = 0$, set the neighborhood size parameter to the small size, go to *Step B*; if the algorithm is in the large neighborhood stage and $0 < q_{tabu} \le \alpha$, go to *Step B*. For all other cases, go to *Step A*.

Fig. 9. *Step C* of the tabu search procedure.

in the literature, in this tabu search algorithm, a move will not be accepted if it is forbidden in the tabu list unless its tabu status is overridden by the aspiration criterion whenever this move results in a better solution than the best found thus far.

4. Computational experiments

The tabu search algorithm described in this paper was examined on 320 test problems published in the literature [2,17,40,13]. The results from the tabu search were compared with those produced by Chao et al.'s 5-step algorithm and a version of Tsiligirides' stochastic heuristic for the STSP that was extended for the TOP by Chao et al. [2]. This tabu search heuristic was implemented in C++ and run on DEC AlphaServer 1200/533 and DEC Alpha XP1000 computers with 1 gigabyte ram and 1.5 gigabytes swap, each running Digital 4.0E operating system, using Digital's C++ compiler.

4.1. Description of the test problems

There are six data sets in the test problems, each with more than 30 vertices: $n = 32$ (data set 1), 33 (data set 3), 100 (data set 4), 66 (data set 5), 64 (data set 6) and 102 (data set 7). All test problems are provided in Chao [40]. Data sets 1 and 3 originated from Tsiligirides' [13] work for

the OP. With the exception of data sets 5 and 6, the locations of vertices in these data sets were randomly generated. In data sets 5 and 6, the locations of the vertices take on a square shape and a diamond shape, respectively, and vertices further away from the starting and ending points have larger rewards than those closer in (see [40] for more information).

4.2. Computational results

The computational results of the tabu search algorithm on the 320 test problems were compared with published results obtained by other heuristics on the same problems:

Tabu: The tabu search heuristic described in this paper;
CGW: Chao et al.'s 5-step heuristic [2];
TSA: A version of Tsiligirides' heuristic extended for the TOP by Chao et al. [2].

As is common in most metaheuristic approaches, we set the parameters used in the algorithm before running the experiments. There are two sets of parameters employed in this tabu search algorithm: one for the small neighborhood stage and the other for the large neighborhood stage, both required in solution improvement (*Step B* of Fig. 2). These two sets of parameters used in the experiments for randomly generated problems (datasets 1, 3, 4 and 7) are as follows:

$$(\alpha_1, \beta_1, \mathrm{T}, \chi_1, \delta_1, \varepsilon_1, \kappa_1, \theta_{\min}, \theta_{\max}) = (n, 2n, 30, 4, 6, 0.02, 5, 5, 10),$$

$$(\alpha_2, \beta_2, \mathrm{T}, \chi_2, \delta_2, \varepsilon_2, \kappa_2, \theta_{\min}, \theta_{\max}) = (n, 8n, 30, 4, 6, 0.05, 7, 6, 12)$$

while the parameters for the square-shaped and diamond-shaped problems are:

$$(\alpha_1, \beta_1, \mathrm{T}, \chi_1, \delta_1, \varepsilon_1, \kappa_1, \theta_{\min}, \theta_{\max}) = (n, 2n, 30, 4, 6, 0, 5, 5, 10),$$

$$(\alpha_2, \beta_2, \mathrm{T}, \chi_2, \delta_2, \varepsilon_2, \kappa_2, \theta_{\min}, \theta_{\max}) = (n, 8n, 30, 4, 6, 0, 7, 6, 12).$$

These parameters were determined based on a small number of preliminary runs. This tabu search heuristic is embedded in the AMP procedure, where parameters $\lambda = 3$, $N = 1500$ and evaluation function (10a) are used, as described in Section 3.2.2. Other values for λ and N may be chosen. Generally, the tabu search heuristic with larger λ and N values will likely obtain better solutions, but with increased computational effort. Note that the only difference in the two parameter sets for computational experiments is parameter ε. By setting a positive ε with value greater than zero, neighborhood solutions generated in the tabu search may be infeasible. It was observed during preliminary experiments that allowing infeasibility was more useful for random problem sets (i.e. problem sets 1, 3, 4 and 7) than it was for other problem sets (i.e. problem sets 5 and 6). This led to the decision to employ different parameter settings for each of the two problem classes.

Recall from the algorithm description given in Section 3, the proposed algorithm alternates between small and large neighborhood stages in solution improvement, which is different from many other tabu search implementations (e.g. [34]). To confirm that alternating between these two neighborhood stages can reduce computation time without degrading solution quality, a comparison tabu search procedure was coded that utilized the same parameters (two sets, one for the small neighborhood stage and the other for the large neighborhood stage) as the tabu search algorithm described in this paper, but separated the small and large neighborhood stages in solution improvement. That is, for the comparison tabu search, the search process enters the large neighborhood stage when no

Table 1
Results for test runs

n	m	L	Tabu		Comparison Tabu	
			Reward	c.p.u.[a]	Reward	c.p.u.[a]
102	4	100.0	1063	148.3	1067	196.9
		95.0	1022	164.1	1021	230.4
		90.0	966	197.6	964	262.3
		85.0	905	184.7	900	209.1
		80.0	832	149.7	838	191.4
Average			**957**	**168.9**	**958**	**218.0**
Average difference from Tabu					0.1%	29.1%
100	4	60.0	1255	243.5	1254	445.9
		57.5	1203	160.4	1221	457.1
		55.0	1165	144.9	1159	410.7
		52.5	1127	330.8	1108	238.5
		50.0	1056	247.2	1094	203.0
Average			**1161**	**225.4**	**1167**	**351.0**
Average difference from Tabu					0.5%	55.7%

[a]DEC AlphaServer 1200/533 computer (seconds).

improved solutions can be found in the small neighborhood stage and does not return to the small neighborhood stage until a new initial solution is generated and the tabu search is restarted, as was implemented by [34].

In Table 1, the two implementations are compared based on the results of test runs on select problems from data sets 4 and 7. These selected problems have comparatively large m and L values, and thus, are likely to be more computationally demanding than other problems with smaller values of m and L in the same data sets. The results for the comparison tabu search procedure are recorded under the column "Comparison Tabu". We see from the test runs that for the comparable average solution values ("Comparison Tabu" only improves average rewards by 0.1% and 0.5%, respectively), the implementation with the separated small and large neighborhood stages has incurred significantly more computational time (increase of 29.1% and 55.7%, respectively). This confirms the conjecture that alternating between the small and large neighborhood stages can be an efficient way of implementing tabu search algorithms without significant loss in solution quality.

Average solution values for the test problems employing this more efficient implementation with respect to changes in the neighborhood size are summarized in Table 2. These average values were obtained based on the results of a single pass of the proposed tabu search algorithm with fixed parameters and compared with the results from Chao et al. [2]. For example, there are 17 problems, each with a different L for data set 4 ($n = 100$) with four vehicles (i.e. $m = 4$). Thus, the average results (reward = 785 for Tabu, 766 for CGW and 713 for TSA) provided in Table 2 are average values on the solutions for these 17 problems. Average solution values in Table 2 are rounded to

Table 2
Summary of one-run results

Set	n	m	Tabu		CGW		TSA		Tabu vs. CGW	
			Reward	c.p.u.$_{max}$ [b]	Reward	c.p.u.$_{max}$ [c]	Reward	c.p.u.$_{max}$ [c]	+	−
1[a]	32	4	**138**	1.5	130	6.8	133	96.9	3	0
3[a]	33	4	**353**	0.8	337	2.6	340	119.3	3	0
4	100	4	**785**	136.8	766	674.9	713	1358.0	13	2
5	66	4	**699**	22.6	696	141.2	675	454.4	7	3
6	64	4	713	19.9	**716**	109.6	624	295.0	1	1
7	102	4	**515**	101.0	497	482.9	481	1059.9	13	1
1[a]	32	3	166	2.6	**170**	2.9	168	120.8	1	3
3[a]	33	3	**634**	3.3	614	9.9	606	163.3	4	1
4	100	3	**844**	317.4	815	707.3	773	1563.0	15	2
5	66	3	**776**	51.7	**776**	157.8	762	516.1	7	4
6	64	3	787	37.2	**788**	135.7	707	336.7	1	2
7	102	3	**593**	143.2	586	557.6	554	1191.5	13	3
1[a]	32	2	**135**	1.3	130	4.0	130	84.3	1	0
3[a]	33	2	**441**	6.6	433	15.4	438	164.8	6	2
4	100	2	**895**	796.7	876	934.8	844	1748.2	16	3
5	66	2	887	71.3	**891**	193.7	886	566.9	8	8
6	64	2	**818**	53.8	815	150.1	746	420.2	3	1
7	102	2	**634**	432.6	**634**	841.4	587	1333.3	9	5

[a]Average based on the results in Tables 3 and 4.
[b]DEC Alpha XP1000 Computer (s).
[c]SUN 4/370 Workstation (s, [2]).

the nearest integers, similar to the rounding used in presenting similar results in Chao et al. [2]. Detailed solution values for these problems are provided in Tables 3–8 (Appendix A).

As is shown in Table 2, the tabu search heuristic provides best average solutions in 14 out of 18 problem categories. For the remaining four problem categories, the tabu search procedure obtained close to the best average solution values. The CGW heuristic provides six best average solutions (two tied in average value with the tabu search heuristic). None of the average solutions of the TSA heuristic is ranked best, as shows that the TSA heuristic is inferior to both the CGW heuristic and the proposed tabu search heuristic. More specifically, out of the 12 random problem categories (i.e. where vertex locations and rewards are randomly assigned), the proposed tabu search procedure obtained 11 best average solution values (the only exception is for the smallest problem with 32 vertices and three vehicles). The results in Table 2 clearly show that the tabu search heuristic described in this paper outperforms the CGW and TSA heuristics in solution quality, especially for randomly generated problem cases.

Note that better solutions can sometimes be obtained through sensitivity analysis (where the parameter settings are varied). The corresponding best results for five of the largest problems in each category with $n \geqslant 64$ obtained during this process are reported in Table 9 in Appendix B. Of all 60 problem instances, the proposed tabu search heuristic obtained 55 best-known results, of which only 12 solution values are tied with the previous best results. Some of the best solutions found

Table 3
Results for data set 1

n	m	L	Tabu		CGW		TSA		Tabu vs. CGW
			Reward	c.p.u.[a]	Reward	c.p.u.[b]	Reward	c.p.u.[b]	
32	4	18.8	**175**	1.5	160	6.8	165	96.9	+
		18.2	**165**	1.3	160	1.2	160	95.1	+
		12.5	**75**	0.8	70	0.5	75	55.4	+
	3	25.0	**220**	1.5	215	2.7	215	120.8	+
		24.3	205	2.6	**215**	2.8	210	117.8	−
		21.7	170	1.4	**175**	2.9	170	100.2	−
		13.3	70	0.8	**75**	1.0	**75**	60.4	−
	2	23.0	**135**	1.3	130	4.0	130	84.3	+

[a]DEC Alpha XP1000 Computer (S).
[b]SUN 4/370 Workstation (s, [2]).

Table 4
Results for data set 3

n	m	L	Tabu		CGW		TSA		Tabu vs. CGW
			Reward	c.p.u.[a]	Reward	c.p.u.[b]	Reward	c.p.u.[b]	
33	4	22.5	**560**	0.7	530	2.6	520	119.3	+
		15.0	**310**	0.8	300	1.2	310	89.6	+
		10.0	**190**	0.6	180	1.9	190	62.6	+
	3	36.7	**750**	3.3	720	3.4	710	163.3	+
		31.7	**680**	3.1	630	6.4	630	149.7	+
		30.0	**640**	2.1	620	3.4	610	136.3	+
		28.3	**590**	2.0	580	9.9	570	134.5	+
		25.0	510	2.0	**520**	7.8	510	122.0	−
	2	47.5	**760**	5.4	750	14.3	740	164.8	+
		42.5	**690**	6.6	680	15.4	680	150.1	+
		30.0	**490**	1.5	470	2.4	**500**	117.2	+
		27.5	**460**	3.8	440	5.8	**460**	105.6	+
		25.0	**410**	3.1	390	2.5	400	100.4	+
		20.0	290	1.2	**300**	3.5	290	84.5	−
		17.5	250	0.8	**260**	1.9	250	75.0	−
		12.5	**180**	1.2	170	0.8	**180**	63.4	+

[a]DEC Alpha XP1000 Computer (S).
[b]SUN 4/370 Workstation (s, [2]).

Table 5
Results for data set 4

n	*m*	*L*	Tabu		CGW		TSA		Tabu vs. CGW
			Reward	c.p.u.[a]	Reward	c.p.u.[b]	Reward	c.p.u.[b]	
100	4	60.0	**1255**	136.8	1253	435.4	1160	1358.0	+
		57.5	**1243**	86.6	1230	674.9	1122	1298.1	+
		55.0	**1165**	79.7	1155	345.9	1061	1231.3	+
		52.5	**1124**	115.5	1084	282.7	1011	1157.8	+
		50.0	**1056**	134.5	996	449.2	968	1089.7	+
		47.5	**1014**	112.8	995	427.1	913	1032.7	+
		45.0	**977**	111.9	932	324.8	891	967.4	+
		42.5	**910**	78.5	895	313.3	807	897.0	+
		40.0	**875**	96.2	847	116.3	773	839.1	+
		37.5	**819**	80.9	770	208.1	714	752.9	+
		35.0	**732**	63.8	697	197.6	685	681.5	+
		32.5	627	47.3	**641**	82.3	575	604.2	−
		30.0	**554**	31.4	545	73.3	504	509.7	+
		27.5	453	23.7	**460**	16.8	419	458.9	−
		25.0	**315**	11.2	304	10.3	299	396.1	+
		22.5	**182**	3.2	**182**	4.8	181	334.1	
		20.0	**38**	1.4	**38**	0.1	**38**	270.5	
	3	80.0	**1288**	241.1	1285	239.8	1185	1563.0	+
		76.7	**1282**	317.4	1239	361.8	1192	1509.3	+
		73.3	**1265**	220.9	1225	467.9	1168	1451.6	+
		70.0	**1249**	210.9	1222	707.3	1139	1384.9	+
		66.7	**1218**	296.2	1115	424.9	1054	1310.8	+
		63.3	**1151**	193.7	1078	434.1	1028	1244.0	+
		60.0	**1119**	143.9	1018	425.6	1000	1173.3	+
		56.7	**1005**	167.3	956	485.8	912	1102.2	+
		53.3	**951**	137.0	946	482.7	899	1039.7	+
		50.0	**906**	164.9	889	310.3	840	967.7	+
		46.7	**860**	169.4	829	305.6	808	902.2	+
		43.3	785	92.3	**798**	360.1	725	822.9	−
		40.0	709	134.1	**717**	205.7	673	756.4	−
		36.7	**646**	50.8	623	145.0	598	661.9	+
		33.3	**579**	43.2	552	161.2	516	566.9	+
		30.0	**465**	56.5	432	65.5	418	473.6	+
		26.7	**333**	22.3	**333**	25.0	292	384.0	
		23.3	**192**	15.3	191	6.1	**192**	341.6	+
		20.0	**38**	1.4	**38**	0.1	**38**	270.6	
	2	120.0	**1306**	457.1	1299	504.2	1218	1748.2	+
		115.0	**1294**	796.7	1286	669.9	1209	1717.9	+
		110.0	**1277**	579.3	1199	237.1	1190	1652.6	+
		105.0	**1255**	766.9	1242	633.0	1161	1603.9	+
		100.0	**1208**	650.3	1199	639.2	1154	1538.4	+
		95.0	**1175**	384.2	1147	899.2	1127	1477.3	+
		90.0	**1150**	317.1	1112	934.8	1073	1413.1	+

Table 5 (*continued*)

n	*m*	*L*	Tabu		CGW		TSA		Tabu vs. CGW
			Reward	c.p.u.[a]	Reward	c.p.u.[b]	Reward	c.p.u.[b]	
		85.0	**1089**	464.2	1039	775.8	1010	1347.6	+
		80.0	**1022**	317.1	1003	669.9	964	1281.9	+
		75.0	**963**	342.4	932	398.4	954	1211.8	+
		70.0	**914**	232.6	899	457.9	901	1140.7	+
		65.0	**915**	332.8	858	628.7	807	1057.2	+
		60.0	**827**	173.7	807	653.1	767	977.1	+
		55.0	**749**	114.0	737	351.9	680	911.9	+
		50.0	666	163.3	**669**	378.2	619	819.0	−
		45.0	**593**	92.3	580	276.8	577	734.5	+
		40.0	517	72.1	**531**	148.9	518	636.8	−
		35.0	438	58.9	**440**	46.3	432	526.5	−
		30.0	**341**	90.2	**341**	37.1	325	417.7	
		25.0	**202**	33.7	194	6.9	197	341.2	+

[a]DEC Alpha XP1000 Computer (S).
[b]SUN 4/370 Workstation (s, [2]).

by the proposed tabu search heuristic significantly improve the previous best results (e.g. for the problem in data set 4 with $m = 4$ and $L = 50.0$, the previous best solution has been improved 9.8%). Solutions obtained by the proposed tabu search procedure are robust, as is evidenced by the fact that there is very small variation (less than 2% in most cases) between the best results (Table 9) and the one-run results (Tables 3–8). Little variation in solution quality was also observed in these experiments when compared across parameter settings.

The c.p.u. times reported under column "$cpu_{\max}$" in Table 2 are the maximum c.p.u. times in seconds of all the problems in each problem category. A direct analysis of running times is difficult, because various methods were implemented using different programming languages and tested on different computers. However, since the DEC Alpha machine is considered significantly faster than Sun 4/370, the proposed tabu search heuristic might be more time-consuming than the CGW heuristic. In a related study [41], modifications were proposed that significantly reduced the computational requirements of an extension of this tabu search heuristic with very small sacrifice in solution quality.

5. Conclusions

A tabu search heuristic embedded in an adaptive memory procedure for the TOP is described in this paper. Several features are incorporated in the proposed tabu search heuristic: (1) the AMP and its mechanism for updating stored solutions allow a comparatively large pool of good and diversified solutions to be stored and used during the search process; (2) alternating between small and large neighborhood stages during the tabu search course enables the search to evolve in an efficient way without significant degradation in solution quality; and (3) both random and

Table 6
Results for data set 5

n	*m*	*L*	Tabu		CGW		TSA		Tabu vs. CGW
			Reward	c.p.u.[a]	Reward	c.p.u.[b]	Reward	c.p.u.[b]	
66	4	32.5	**1575**	12.5	1545	137.8	1495	454.4	+
		31.2	**1520**	22.6	1490	113.1	1445	437.9	+
		30.0	1410	13.3	**1420**	96.7	1410	421.7	−
		28.8	**1380**	14.1	**1380**	141.2	1330	401.3	
		27.5	**1310**	19.1	**1310**	116.4	1310	378.3	
		26.2	**1275**	13.1	1260	93.3	1255	369.4	+
		25.0	1100	7.9	**1160**	73.8	1100	336.9	−
		23.8	1000	14.6	**1020**	69.0	995	323.8	−
		22.5	**960**	8.5	950	84.2	890	289.2	+
		21.2	**860**	7.4	**860**	69.9	795	275.1	
		20.0	**760**	6.8	750	102.4	715	256.7	+
		18.8	**680**	5.9	675	80.4	655	235.7	+
		17.5	**620**	5.7	**620**	48.6	585	220.4	
		16.2	**555**	5.9	495	47.4	525	194.2	+
		15.0	**430**	4.9	**430**	43.4	410	170.5	
		13.8	**340**	5.0	**340**	25.7	335	161.3	
		12.5	**340**	2.9	**340**	4.5	315	149.0	
		11.2	**240**	2.3	**240**	31.0	210	129.3	
		10.0	**140**	1.8	**140**	17.9	**140**	129.8	
		8.8	**140**	1.6	**140**	2.7	**140**	123.3	
		7.5	**80**	1.4	**80**	30.8	**80**	108.6	
		6.2	**20**	0.1	**20**	0.1	**20**	87.1	
		5.0	**20**	0.1	**20**	0.1	**20**	87.2	
		3.8	**20**	0.3	**20**	0.1	**20**	87.2	
	3	43.3	**1635**	51.7	**1635**	140.3	1595	516.1	
		41.7	**1580**	29.2	1570	157.8	1550	501.7	+
		40.0	**1530**	26.0	**1530**	117.0	1485	488.9	
		38.3	**1465**	26.8	1450	135.9	1430	461.3	+
		36.7	**1410**	26.0	1400	141.9	1385	457.1	+
		35.0	**1330**	39.1	**1330**	133.5	1305	427.4	
		33.3	1240	19.9	**1250**	144.0	1235	396.8	−
		31.7	**1175**	19.5	**1175**	139.7	1175	386.5	
		30.0	**1115**	25.0	1105	132.6	1105	355.4	+
		28.3	**1065**	17.2	1060	109.8	1040	336.1	+
		26.7	**990**	11.5	**990**	101.8	985	312.2	
		25.0	835	10.7	**870**	82.6	840	288.1	−
		23.3	**755**	9.6	**755**	100.0	755	264.5	
		21.7	**650**	8.3	**650**	70.0	650	241.3	
		20.0	575	7.8	**595**	86.8	545	220.8	−
		18.3	**495**	7.5	480	66.6	480	196.0	+
		16.7	**470**	5.7	**470**	27.1	445	174.1	
		15.0	**335**	6.3	**335**	36.2	295	147.3	
		13.3	**260**	6.0	255	3.6	255	139.5	+
		11.7	**185**	8.0	**185**	18.8	175	118.1	

Table 6 (*continued*)

n	*m*	*L*	Tabu		CGW		TSA		Tabu vs. CGW
			Reward	c.p.u.[a]	Reward	c.p.u.[b]	Reward	c.p.u.[b]	
		10.0	**110**	5.5	**110**	31.7	**110**	116.6	
		8.3	95	1.7	**110**	2.1	**105**	108.5	−
		6.7	**60**	1.9	**60**	19.1	**60**	95.6	
		5.0	**20**	0.1	**20**	0.2	**20**	86.6	
		3.3	**15**	1.1	**15**	7.8	**15**	83.2	
	2	65.0	1665	63.2	**1680**	81.2	1670	566.9	−
		62.5	1630	53.9	**1635**	123.6	1635	552.9	−
		60.0	**1610**	71.3	1600	144.0	1610	548.0	+
		57.5	**1560**	52.0	1545	155.4	1550	523.5	+
		55.0	**1500**	31.4	1490	185.2	1500	527.9	+
		52.5	1445	36.5	**1450**	154.2	1450	492.6	−
		50.0	**1380**	34.9	**1380**	180.9	1375	487.4	
		47.5	**1310**	33.9	**1310**	193.7	1320	453.0	
		45.0	**1260**	46.4	1250	147.3	1250	428.9	+
		42.5	1185	51.4	**1195**	155.1	1190	409.6	−
		40.0	1090	53.6	**1150**	163.6	1150	390.5	−
		37.5	975	30.1	**1010**	95.6	1010	367.1	−
		35.0	**920**	21.6	**920**	123.4	920	341.0	
		32.5	**860**	23.8	855	122.6	850	315.0	+
		30.0	770	18.3	**790**	102.5	750	282.5	−
		27.5	**670**	13.8	**670**	85.6	660	260.8	
		25.0	560	11.7	**580**	33.3	580	234.2	−
		22.5	**480**	12.2	**480**	39.8	480	208.0	
		20.0	**410**	10.8	395	63.6	375	184.0	+
		17.5	**320**	7.6	315	8.4	310	165.8	+
		15.0	**240**	5.3	**240**	41.6	200	128.3	
		12.5	**180**	3.9	175	3.7	170	119.8	+
		10.0	**80**	2.8	**80**	11.1	75	102.3	
		7.5	**50**	1.6	**50**	9.2	40	89.9	
		5.0	**20**	1.0	**20**	0.3	20	87.5	

[a]DEC Alpha XP1000 Computer (S).
[b]SUN 4/370 Workstation (s, [2]).

greedy procedures are employed in neighborhood solution exploration. These features are largely responsible for the efficiency and effectiveness of this tabu search algorithm in the computational experiments.

The structure of the proposed tabu search is flexible, allowing for further improvements and the use of new extensions. For example, one could use a solution procedure that addresses the set packing problem as a post-optimization technique to improve solutions that are produced by the tabu search procedure (see, e.g. [42]). In addition, no significant revision in neighborhood structure would be necessary for the current algorithm to consider more realistic problem features

Table 7
Results for data set 6

n	m	L	Tabu		CGW		TSA		Tabu vs. CGW
			Reward	c.p.u.[a]	Reward	c.p.u.[b]	Reward	c.p.u.[b]	
64	4	20.0	**1068**	8.9	**1068**	109.6	936	295.0	
		18.8	**912**	11.0	**912**	69.5	738	251.5	
		17.5	**696**	8.7	690	85.1	624	215.9	+
		16.2	522	4.4	**546**	43.9	504	220.9	−
		15.0	**366**	3.2	**366**	26.4	318	140.5	
	3	26.7	1152	18.6	**1158**	134.4	1086	336.7	−
		25.0	**1080**	15.4	**1080**	135.7	954	309.4	
		23.3	**990**	14.5	972	119.6	888	271.4	+
		21.7	876	10.8	**894**	79.1	768	247.7	−
		20.0	**828**	10.1	**828**	109.8	732	235.5	
		18.3	612	15.0	**642**	87.1	594	232.8	−
		16.7	**444**	7.5	**444**	58.9	396	154.9	
		15.0	**282**	3.5	**282**	35.8	240	118.0	
	2	40.0	**1260**	45.7	1242	149.0	1158	420.2	+
		37.5	**1188**	30.3	1176	150.1	1086	385.4	+
		35.0	**1116**	26.3	1104	127.3	1020	354.2	+
		32.5	**1032**	20.0	**1032**	137.9	954	323.9	
		30.0	936	17.6	**942**	131.6	852	279.8	−
		27.5	**888**	16.7	**888**	110.1	804	252.2	
		25.0	**780**	14.1	**780**	101.4	690	226.2	
		22.5	**660**	12.1	**660**	90.3	594	190.6	
		20.0	**588**	11.1	**588**	59.0	540	176.9	
		17.5	**360**	8.3	**360**	43.1	336	134.6	
		15.0	**192**	4.5	**192**	26.4	174	97.0	

[a]DEC Alpha XP1000 Computer (S).
[b]SUN 4/370 Workstation (s, [2]).

such as time-dependent customer rewards and stochastic service or travel times. By employing a less vigorous set of parameters and providing guidance criteria for vertex selection in the algorithm, the search process will be able to tackle some large size problems. In a related work [41], this tabu search procedure was modified to solve a real-world application with more than 2000 vertices.

Acknowledgements

This work was supported by NSF grant CMS 9875305 and by United Technologies Research Center of United Technologies Corporation. This support is gratefully acknowledged, but implies no endorsement of the findings.

Table 8
Results for data set 7

n	*m*	*L*	Tabu		CGW		TSA		Tabu vs. CGW
			Reward	c.p.u.[a]	Reward	c.p.u.[b]	Reward	c.p.u.[b]	
102	4	100	**1067**	86.6	1066	482.9	980	1059.9	+
		95	**1019**	84.3	990	466.1	940	1000.1	+
		90	**966**	101.0	886	430.2	889	944.7	+
		85	**905**	95.2	882	336.1	851	892.2	+
		80	**832**	82.0	801	332.6	765	830.7	+
		75	**776**	71.3	728	269.3	712	783.2	+
		70	**726**	54.4	683	278.1	644	720.4	+
		65	**643**	68.6	610	175.9	606	663.5	+
		60	**576**	31.8	562	63.8	543	584.2	+
		55	503	44.9	**516**	103.6	476	538.8	−
		50	**462**	23.6	**462**	26.7	441	485.5	
		45	**359**	20.6	338	56.9	351	437.1	+
		40	**285**	17.2	283	10.6	283	393.7	+
		35	**217**	10.0	209	7.9	211	358.9	+
		30	**164**	5.5	156	4.1	161	332.4	+
		25	**123**	3.3	**123**	3.7	**123**	312.1	
		20	**79**	0.1	**79**	0.7	**79**	294.5	
		15	**46**	0.1	**46**	0.1	**46**	280.3	
		10	**30**	0.1	**30**	0.1	**30**	272.4	
	3	133.3	**1098**	143.2	1095	557.6	1025	1191.5	+
		126.7	1061	99.8	**1064**	498.3	954	1125.6	−
		120.0	**1011**	93.6	1008	443.7	931	1079.5	+
		113.3	**966**	98.8	943	369.7	911	1022.1	+
		106.7	**922**	74.0	919	397.7	885	966.6	+
		100.0	**874**	102.8	848	457.3	805	914.5	+
		93.3	789	126.5	**813**	254.8	742	856.5	−
		86.7	**756**	121.2	754	317.5	698	792.8	+
		80.0	**681**	69.5	676	243.7	635	727.8	+
		73.3	**632**	94.4	602	255.7	585	670.8	+
		66.7	**563**	107.7	539	178.9	537	598.9	+
		60.0	**481**	36.0	466	57.9	445	525.0	+
		53.3	416	34.0	**419**	76.7	388	472.3	−
		46.7	**344**	21.1	338	13.2	322	422.9	+
		40.0	**247**	23.7	235	18.0	**247**	368.7	+
		33.3	**175**	12.0	163	4.7	**175**	330.0	+
		26.7	**117**	3.7	**117**	4.8	**117**	313.0	
		20.0	**79**	2.3	**79**	0.8	**79**	291.7	
		13.3	**46**	0.1	**46**	0.1	**46**	280.1	
	2	200	1165	290.6	**1173**	789.6	1045	1333.3	−
		190	1116	215.6	**1127**	755.8	1026	1285.3	−
		180	1067	432.6	**1082**	841.4	971	1231.7	−
		170	1017	239.6	**1031**	690.9	964	1184.3	−
		160	**987**	272.1	**987**	818.4	904	1127.7	
		150	914	202.8	**934**	542.8	867	1069.9	−

Table 8 (*continued*)

n	*m*	*L*	Tabu		CGW		TSA		Tabu vs. CGW
			Reward	c.p.u.[a]	Reward	c.p.u.[b]	Reward	c.p.u.[b]	
		140	**864**	224.3	**864**	445.5	788	1014.9	
		130	**817**	174.1	811	412.2	751	962.2	+
		120	**767**	217.5	758	272.5	698	916.8	+
		110	**702**	120.1	693	284.1	644	837.4	+
		100	**638**	118.7	633	377.5	591	771.3	+
		90	**578**	84.4	576	231.4	539	695.3	+
		80	**521**	52.0	517	326.8	497	621.3	+
		70	**459**	74.6	453	173.2	436	546.4	+
		60	**382**	42.7	379	99.2	360	462.9	+
		50	**290**	37.7	275	40.9	277	389.8	+
		40	**190**	16.3	**190**	16.1	179	343.7	
		30	**101**	8.6	**101**	12.1	**101**	300.6	
		20	**64**	2.8	**64**	2.7	**64**	288.3	
		10	**30**	0.1	**30**	0.4	**30**	272.7	

[a]DEC Alpha XP1000 Computer (S).
[b]SUN 4/370 Workstation (s, [2]).

Table 9
Summary of best results

n	*m*	*L*	Tabu Best	Previous best
100	4	60.0	**1270**	1253
(Date set 4)		57.5	**1252**	1230
		55.0	**1197**	1155
		52.5	**1141**	1084
		50.0	**1094**	996
	3	80.0	**1296**	1285
		76.7	**1282**	1239
		73.3	**1269**	1225
		70.0	**1250**	1222
		66.7	**1220**	1115
	2	120	**1306**	1299
		115	**1298**	1286
		110	**1277**	1199
		105	**1255**	1242
		100	**1208**	1199
66	4	32.5	**1610**	1545
(Data set 5)		31.2	**1520**	1490
		30.0	**1420**	**1420**

Table 9 (*continued*)

n	*m*	*L*	Tabu Best	Previous best
		28.8	**1380**	**1380**
		27.5	**1320**	1310
	3	43.3	**1635**	**1635**
		41.7	**1590**	1570
		40.0	**1555**	1530
		38.3	**1465**	1450
		36.7	**1420**	1400
	2	65.0	1670	**1680**
		62.5	**1635**	**1635**
		60.0	**1610**	1600
		57.5	**1560**	1545
		55.0	**1500**	1490
64	4	20.0	**1068**	**1068**
(Data Set 6)		18.8	**912**	**912**
		17.5	**696**	690
		16.2	528	**546**
		15.0	**366**	**366**
	3	26.7	**1170**	1158
		25.0	**1080**	**1080**
		23.3	**990**	972
		21.7	**894**	**894**
		20.0	**828**	**828**
	2	40.0	**1260**	1242
		37.5	**1188**	1176
		35.0	**1116**	1104
		32.5	**1032**	**1032**
		30.0	**942**	**942**
102	4	100.0	**1077**	1066
(Data set 7)		95.0	**1019**	990
		90.0	**966**	889
		85.0	**905**	882
		80.0	**838**	801
	3	133.3	**1110**	1095
		126.7	**1065**	1064
		120.0	**1013**	1008
		113.3	**984**	943
		106.7	**922**	919
	2	200.0	1170	**1173**
		190.0	1123	**1127**
		180.0	1072	**1082**
		170.0	**1036**	1031
		160.0	**988**	987

Appendix A.

A.1. One-run results of the tabu search heuristic vs. other heuristics

Tables 3–8 summarize the results of one pass with fixed parameters of the proposed tabu search heuristic given in Section 3. Solution values in bold are the best results for each problem category. Column headings in these tables are also defined previously.

Appendix B.

B.1. Summary of the best results

Table 9 summarizes the best results for selected problem instances. In column *Tabu Best*, best solution values obtained by changing parameters for the proposed tabu search heuristic are reported. Column *Previous Best* summarizes the best solution values reported in the literature. Other column headings are previously defined. The best values by any technique are shown in bold.

References

[1] Golden B, Levy L, Vohra R. The orienteering problem. Naval Research Logistics 1987;34:307–18.
[2] Chao I, Golden B, Wasil E. The team orienteering problem. European Journal of Operational Research 1996a;88: 464–74.
[3] Butt S, Cavalier T. A heuristic for the multiple tour maximum collection problem. Computers and Operations Research 1994;21:101–11.
[4] Ballou R, Chowdhury M. MSVS: an extended computer model for transport mode selection. The Logistics and Transportation Review 1980;16:325–38.
[5] Diaby M, Ramesh R. The distribution problem with carrier service: a dual based penalty approach. ORSA Journal on Computing 1995;7:24–35.
[6] Hall R, Racer M. Transportation with common carrier and private fleets: system assignment and shipment frequency optimization. IIE Transactions 1995;27:217–25.
[7] Dueck G, Scheuer T. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. Journal of Computational Physics 1990;90:161–75.
[8] Butt S, Ryan D. An optimal solution procedure for the multiple tour maximum collection problem using column generation. Computer and Operations Research 1999;26:427–41.
[9] Laporte G, Martello S. The selective traveling salesman problem. Discrete Applied Mathematics 1990;26:193–207.
[10] Ramesh R, Yoon Y, Karwan M. An optimal algorithm for the orienteering tour problem. ORSA Journal on Computing 1992;4:155–65.
[11] Fischetti M, Salazar J, Toth P. Solving the orienteering problem through branch-and-cut. INFORMS Journal on Computing 1998;10:33–148.
[12] Gendreau M, Laporte G, Semet F. A branch-and-cut algorithm for the undirected selective traveling salesman problem. Networks 1998b;32:263–73.
[13] Tsiligirides T. Heuristic methods applied to orienteering. Journal of the Operational Research Society 1984;35: 797–809.

[14] Wren A, Holliday A. Computer scheduling of vehicles from one or more depots to a number of delivery points. Operational Research Quarterly 1972;23:333–44.
[15] Golden B, Wang Q, Liu L. A multifaceted heuristic for the orienteering problem. Naval Research Logistics 1988;35:359–66.
[16] Ramesh R, Brown K. An efficient four-phase heuristic for the generalized orienteering problem. Computers and Operations Research 1991;18:151–65.
[17] Chao I, Golden B, Wasil E. A fast and effective heuristic for the orienteering problem. European Journal of Operational Research 1996b;88:475–89.
[18] Gendreau M, Laporte G, Semet F. A tabu search heuristic for the undirected selective traveling salesman problem. European Journal of Operational Research 1998a;10:539–45.
[19] Fischetti M, Toth P. An additive approach for the optimal solution of the prize-collecting traveling salesman problem. In: Golden B, Assad A, editors. Vehicle routing: methods and studies. Amsterdam: Elsevier Science Publishers; 1988. p. 319–43.
[20] Larson R, Odoni A. Urban operations research. Englewood Cliffs, NJ: Prentice-Hall; 1981.
[21] Golden B, Wasil E, Kelly J, Chao I. The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: Crainic T, Laporte G, editors. Fleet management and logistics. Dordrecht: Kluwer Academic Publishers; 1998. p. 33–56.
[22] Gendreau M, Laporte G, Potvin J-Y. Metaheuristics for the capacitated VRP. In: Toth P, Vigo D, editors. The vehicle routing problem, Philadelphia: Society for Industrial & Applied Mathematics; 2002, p. 129–54.
[23] Glover F. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research 1986;1:533–49.
[24] Glover F. Tabu search: part I. ORSA Journal on Computing 1989;1:190–206.
[25] Glover F. Tabu search: part II. ORSA Journal on Computing 1990;2:4–32.
[26] Golden B, Laporte G, Taillard E. An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. Computers and Operations Research 1997;24:445–52.
[27] Rochat Y, Taillard É. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1995;1:147–67.
[28] Gendreau M, Laporte G, Musaraganyi C, Taillard É. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. Computers and Operations Research 1999;26:1153–73.
[29] Taillard É, Gambardella L, Gendreau M, Potvin J-Y. Adaptive memory programming: a unified view of metaheuristics. European Journal of Operational Research 2001;135:1–16.
[30] Glover F. Tabu search and adaptive memory programming: advances, applications and challenges. In: Barr R, Helgason R, Kennington J, editors. Advances in metaheuristics, optimization and stochastic modeling technologies, Dordrecht: Kluwer Academic Publishers; 1997. p. 1–75.
[31] Taillard É, Badeau P, Gendreau M, Guertin F, Potvin J-Y. A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation science 1997;31:170–86.
[32] Rosenkrantz D, Stearns R, Lewis P. Approximate algorithms for the traveling salesman problem. Proceedings of the 15th annual IEEE symposium of switching and automata theory. 1974. p. 33–42.
[33] Gendreau M, Hertz A, Laporte G. A tabu search heuristic for the vehicle routing problem. Management Science 1994;40:1276–90.
[34] Barbarosoglu G, Ozgur D. A tabu search algorithm for the vehicle routing problem. Computers and Operations Research 1999;26:255–70.
[35] Lin S, Kernighan B. An effective heuristic algorithm for the traveling salesman problem. Operations Research 1973;21:498–516.
[36] Mladenović N, Hansen P. Variable neighborhood search. Computers and Operations Research 1997;24:1097–110.
[37] Liu F-H, Shen S-Y. A route-neighborhood-based metaheuristic for vehicle routing problem with time windows. European Journal of Operational Research 1999;118:485–504.
[38] Hart J, Shogan A. Semi-greedy heuristics: an empirical study. Operations Research Letters 1987;6:107–14.
[39] Shaw P. Using constraint programming and local search methods to solve vehicle routing problems. In: Maher M, Puget J-F, editors. Principles and practice of constraint programming–CP98, Lecture Notes in Computer Science, Berlin: Springer, 1998. p. 417–31.

[40] Chao I. Algorithms and solutions to multi-level vehicle routing problems. PhD Dissertation, University of Maryland, College Park, 1993.
[41] Miller-Hooks E, Tang H. Scheduling of planned maintenance tasks for otis: final report. Report to United Technologies Research Center of United Technologies Corporation, East Hartford, Connecticut, 2002.
[42] Taillard É. A heuristic column generation method for the heterogeneous fleet VRP. Recherche-Operationnelle 1999;33:1–14.