European Journal of Operational Research 229 (2013) 573–584

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

A memetic algorithm for the multiperiod vehicle routing problem with profit

Zizhen Zhang [a], Oscar Che [a], Brenda Cheang [b], Andrew Lim [a], Hu Qin [c,*]

[a] Department of Management Sciences, City University of Hong Kong, Tat Chee Avenue, Kowloon Tong, Kowloon, Hong Kong
[b] Division of Information and Technology Studies, Faculty of Education, The University of Hong Kong, Pokfulam, Hong Kong
[c] School of Management, Huazhong University of Science and Technology, Wuhan, China

A R T I C L E I N F O

Article history:
Received 31 March 2011
Accepted 30 November 2012
Available online 21 December 2012

Keywords:
Memetic algorithm
Metaheuristics
Giant-tour
Multiperiod
Periodic vehicle routing

A B S T R A C T

In this paper, we extend upon current research in the vehicle routing problem whereby labour regulations affect planning horizons, and therefore, profitability. We call this extension the multiperiod vehicle routing problem with profit (mVRPP). The goal is to determine routes for a set of vehicles that maximizes profitability from visited locations, based on the conditions that vehicles can only travel during stipulated working hours within each period in a given planning horizon and that the vehicles are only required to return to the depot at the end of the last period. We propose an effective memetic algorithm with a giant-tour representation to solve the mVRPP. To efficiently evaluate a chromosome, we develop a greedy procedure to partition a given giant-tour into individual routes, and prove that the resultant partition is optimal. We evaluate the effectiveness of our memetic algorithm with extensive experiments based on a set of modified benchmark instances. The results indicate that our approach generates high-quality solutions that are reasonably close to the best known solutions or proven optima, and significantly better than the solutions obtained using heuristics employed by professional schedulers.

1. Introduction

This paper examines a problem faced by a buying office that procures assorted products predominantly from over 1000 direct-source suppliers in Hong Kong and China for one off the largest retail distributions in the world. Upon the placement of product orders by the buying office, the goods must be inspected before shipment. Thus, each supplier has to make an inspection request with the buyer when the ordered goods are ready for delivery. The buyer, in turn, schedules a professional quality inspector to perform an on-site inspection. And in order to facilitate coordination between inspectors and suppliers, the inspections should occur during office hours (e.g. 9:00 am to 5:00 pm). A weekly schedule is created to assign inspectors to requests for the upcoming week. The buying office has a stable of in-house inspectors who receive their weekly inspection schedules at regional offices, and each inspector only reports back to their regional offices after they have completed all their inspections for the week (note that inspectors are not required to return to the regional offices every day). During the week, an inspector generally travels to different locations and performs several inspections per day, and then finds overnight accommodation in the vicinity of his/her last/next inspection site. The objective of the problem is to assign as many inspection requests as possible to the stable of inspectors while satisfying the working hour constraints; unfulfilled inspection requests are outsourced and are appropriated as additional costs.

A variant of the well-studied vehicle routing problem (VRP), we call the model of this problem the *multiperiod vehicle routing problem with profit* (mVRPP). It is defined on a complete undirected graph, where each node represents a location with an associated reward (in the case of inspection scheduling, this is equivalent to the cost of outsourcing that inspection request), and the weight of each edge is the traveling distance between the corresponding locations. The goal is to devise a set of K vehicle routes that maximizes the total reward collected from the visited nodes, and each node can be visited at most once. Each vehicle route is divided into at most D sub-routes (called *trips*), and the length of each trip is limited by working hour constraints. Vehicles depart from the depot at the beginning of the planning horizon. Each vehicle stays at the last visited node at the end of each trip and begins the next trip from that node. Finally, the last trip must end at the depot.

There are two features that distinguish the mVRPP from existing research on routing and scheduling problems. The first is the requirement of having D periods. Many studies on such problems assume that the scheduling subjects (e.g. vehicles or field technicians) are always in service within the planning horizon. However, this assumption may not be valid for many practical applications such as those of the freight and transportation industry. This is

* Corresponding author. Tel.: +852 64117909; fax: +852 34420189.
E-mail addresses: zizzhang@cityu.edu.hk (Z. Zhang), oscarche@cityu.edu.hk (O. Che), lim.andrew@cityu.edu.hk (A. Lim), tigerqin1980@gmail.com (H. Qin).

0377-2217/$ - see front matter © 2012 Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.ejor.2012.11.059

due to working hour regulations, where sufficient downtime for rest and recuperation is essential in terms of road safety for drivers, and providing punctual service deliveries during working hours is always well-appreciated by clients. The second feature is the objective of maximizing total reward rather than the number of vehicles to the total distance travelled.

In this paper, we devise a Memetic Algorithm (MA) (Moscato, 1999) to solve the mVRPP, which is an approach combining an evolutionary algorithm (e.g. genetic algorithm) with a local improvement procedure. MA has been successfully implemented on various routing and scheduling problems, including homogeneous (Prins, 2004) and heterogeneous fleets (Prins, 2009). In our proposed MA, a solution consisting of a set of K routes is represented by a single chromosome called a giant-tour (Prins, 2004), which is simply a permutation of the set of nodes in the problem graph. We describe a fast, exact greedy procedure to partition a given giant-tour into a set of K routes, and show that the resultant solution is optimal for the giant-tour. Essentially, our approach can be classified under the family of memetic algorithms for the VRP proposed by Prins (2004).

The contributions of this study are threefold. First, we introduce a new but practical scheduling problem that considers regulated working hours in multiperiod planning with a profit maximization objective. Second, we provide an effective memetic algorithm (MA) for the problem that incorporates a fast greedy procedure for the optimal decoding of giant-tour chromosomes. Third, our dataset and comprehensive experimental results serve as a baseline for future researchers working on this and related problems.

The remainder of this paper is organised as follows. In Section 2, we briefly describe the relevant literature, including studies concerning working hour regulations and the team orienteering problem (TOP). We then provide a formal definition of the mVRPP in Section 3. In Section 4, we introduce our memetic algorithm approach, which combines a genetic algorithm with a local improvement procedure to solve this problem. To evaluate our approach, Section 5 reports the series of experiments that we conducted based on modified existing benchmark instances for related problems. And with some suggestions for future research in this area, Section 6 closes our study.

2. Related work

Many variations of the vehicle routing problem (VRP) have been studied in existing literature. In most cases, the objective of the VRP variant is to minimize the number of vehicles used and/or the total travel distance. This is different from the mVRPP, whose objective is to maximize the total profit. We refer the reader to Laporte (2007) for an overview of the VRP. In addition, Toth and Vigo (2002) and Golden et al. (2008) provide detailed discussions on the state of the art for the VRP and its future research directions.

One of the defining characteristics of the mVRPP that distinguishes it from other VRP variants is the consideration of working hour regulations. Savelsbergh and Sol (1998) studied a dynamic and generalized pickup and delivery problem in which lunch breaks and night breaks must be taken within fixed time intervals, while Powell et al. (2000) and Campbell and Savelsbergh (2004) considered the maximum number of working hours permitted during a tour as capacity restrictions. Tan et al. (2007) considered regular working hours and overtime as soft constraints. Velasco et al. (2009) studied a pickup and delivery problem that considers the time availability of a helicopter due to the limit on fuel capacity. Recently, some researchers have investigated certain routing and scheduling problems under specific sets of actual working hour regulations. Xu et al. (2003) applied column generation techniques to solve a heterogeneous vehicle pickup-and-delivery problem involving several practical constraints such as nested loading and unloading order constraints on loads, and working hour restrictions by the United States Department of Transportation. By using fast heuristics to solve the sub-problems, they generated a near-optimal solution for several randomly-generated problems. Similarly, Goel (2009), Goel (2010) and Kok et al. (2010) studied a combined vehicle routing and driver scheduling problem under the European Union regulations for drivers. Goel (2009) proposed two scheduling approaches embedded into a large neighbourhood search algorithm; Kok et al. (2010) proposed a dynamic programming algorithm that extends the scheduling approach and considers the ignored regulations in Goel (2009); and Goel (2010) presented a procedure which is always able to find a feasible schedule compliant with the EU regulations if one exists.

Most existing work on routing and scheduling problems with working time considerations concerns truck drivers, which involves different factors compared to our inspector scheduling problem. There is relatively less existing research on the routing and scheduling of field technicians. Bostel et al. (2008) developed a memetic algorithm and a column generation technique for a field technician planning problem over a rolling horizon that considers time windows and lunch breaks, and the technicians are required to start and end their working days at the depot. Qin et al. (2009) proposed a tabu search approach to solve the single vehicle version of the mVRPP.

The other defining characteristic of the mVRPP is its profit maximization objective. This is different from most other VRP variants in literature, whose objective is usually to minimize the number of vehicles used and/or the total travel distance. A previously studied problem with this objective is the team orienteering problem (TOP), which requires the determination of a set of routes maximizing the total reward of nodes visited during a single period with a distance limit; this is a special case of the mVRPP (where the number of periods $D = 1$). Existing research on the TOP has focused on developing meta-heuristic algorithms to solve the problem. The first published TOP heuristic was developed by Chao et al. (1996). Several meta-heuristic algorithms have been subsequently proposed, including tabu search (Tang and Miller-Hooks, 2005), variable neighbourhood search (Archetti et al., 2007), ant colony optimization (Ke et al., 2008), GRASP (Souffriau et al., 2010) and memetic algorithm (Bouly et al., 2010). For further details about the TOP, we refer the reader to the survey by Vansteenwegen et al. (2011).

The multiperiod planning horizon of the mVRPP is related to the periodic vehicle routing problem (PVRP) (Cordeau et al., 1997). In the PVRP, two types of decisions are involved in the planning, namely determining the visiting sequence of each location and the routing plan for each period. Population-based algorithms using a giant-tour representation have been shown to be effective for the PVRP. Mendoza et al. (2009) developed a distance-constrained routing module for a decision support system to resolve an auditor routing problem over multiple periods. The module includes two memetic algorithms to generate routing plans, and two integer-programming clustering models to balance the workload of each auditor over periods. However, the PVRP usually requires each vehicle to return to the depot at the end of each period. This requirement gives the PVRP (and its solution techniques) a very different flavour from the mVRPP. Another problem with a multiperiod planning horizon is the periodic capacitated arc routing problem, which was examined by Chu et al. (2006) who also employed a giant-tour representation in their scatter search approach.

3. Problem description

The *multiperiod vehicle routing problem with profit (mVRPP)* is defined on a complete undirected graph $G = (V,E)$, where

$V = \{0,\ldots,n\}$ is the set of nodes, and $E = \{(i,j)|i,j \in V\}$ is the set of edges. Each node i has an associated reward (or profit) w_i, and an associated service time s_i. The depot node is labelled 0 with reward $w_0 = 0$ and service time $s_0 = 0$. Each edge $(i,j) \in E$ has a non-negative cost c_{ij}, where c_{ij} is the travel time between i and j, and the travel time matrix satisfies the triangle inequality. There are K vehicles that begin at the depot. The objective of the mVRPP is to schedule K routes, each starting and ending at node 0, in the planning horizon consisting of D periods such that the total reward collected from all visited nodes is maximized. We assume that when a vehicle visits a node, the service begins immediately. Furthermore, each node can be visited at most once, and the accumulated travel time in any period for a vehicle cannot exceed a limit L.

Since a vehicle begins service immediately upon arriving at a node, we can convert the graph G into a complete bidirected graph where the cost of each edge is (i,j) is $c_{ij} + s_j$, and solve the equivalent problem; this eliminates the necessity of explicitly considering the service time s_i for each node i. For the remainder of this paper, we employ this alternative problem representation, and assume that the cost of each directed edge c_{ij} includes the service time for node j.

We denote a feasible solution to the mVRPP by S, which is a set of K routes, i.e. $S = \{r_1, r_2, \ldots, r_K\}$. Each route starts and ends at the depot. A route $r_k = (r_k^1, r_k^2, \ldots, r_k^D)$ is divided into D sub-routes called *trips*, where r_k^d is a sequence of nodes representing a trip in period d. We denote the starting and ending nodes of trip r_k^d by $v_s(r_k^d)$ and $v_e(r_k^d)$, respectively. We assume the vehicle stays at the node $v_e(r_k^d)$ at the end of period d, and the vehicle will start a new trip from $v_e(r_k^d)$ in next period, i.e. $v_e(r_k^d) = v_s(r_k^{d+1})$, $d \leqslant D-1$. Note that $v_s(r_k^1) = v_e(r_k^D) = 0$. A vehicle may return to the depot before period D, whereupon it remains at the depot for the rest of the planning horizon.

Fig. 1 illustrates an example of an mVRPP solution $S = \{r_1, r_2\}$ involving $K = 2$ routes and $D = 3$ periods. The route $r_1 = (r_1^1, r_1^2, r_1^3)$ comprises three trips, where $r_1^1 = (0,1,2)$, $r_1^2 = (2,3,4)$ and $r_1^3 = (4,5,0)$. In the first period, the vehicle starts from node 0 and visits nodes 1 and 2 to collect their respective rewards. At the end of the first period, the vehicle stays in node 2. In the second period, the vehicle departs from node 2, visits nodes 3 and 4, and stays at node 4. Finally, in the third period the vehicle departs from node 4, visits node 5 and returns to the depot (node 0).

Let $T(r_k^d)$ be the total travel time for route k in period d. A trip r_k^d is feasible if $T(r_k^d) \leqslant L$, and all non-depot nodes in r_k^d are visited at most once. A route r_k is feasible if all its trips are feasible, and all non-depot nodes in r_k are visited at most once. A solution S is feasible if all its routes are feasible, and all non-depot nodes are visited at most once.

Let $W(S)$ be the total amount of reward collected by solution S. We similarly define $W(r_k^d)$ and $W(r_k)$ to be the collected rewards by trip r_k^d and route r_k, respectively. A solution S is optimal if $W(S)$ is maximum among all feasible solutions.

4. Memetic algorithm

Algorithm 1 presents the overall process of our MA approach to solve the mVRPP. The population P consists of a set of chromosomes, which are initialized with randomly generated permutations at the start of the algorithm. In each generation, we create an offspring population P_{new} as follows. First, we uniformly randomly select two members of P and apply a crossover operation, producing two new chromosomes that we place into P_{new}; this continues until there are *off_size* chromosomes in P_{new}. Next, we perform a mutation operation on each member of $P \cup P_{new}$ with probability ρ. Finally, we apply a local improvement procedure on each chromosome in $P \cup P_{new}$. The best *pop_size* chromosomes from $P \cup P_{new}$ are selected to be the initial population P for the next generation. Whenever a new solution is formed at any stage of the process, it is evaluated and the best solution found is retained. The algorithm runs until *max_iter* consecutive generations have occurred where the best solution found has not been updated.

Algorithm 1. Process of the Memetic Algorithm

$P \leftarrow$ *pop_size* randomly generated permutations;
repeat
  New offspring population $P_{new} \leftarrow \emptyset$;
  repeat
    Uniformly randomly select two parent chromosomes from P;
    Produce two offspring from the parent chromosomes using Crossover;
    Put offspring chromosomes into $\boldsymbol{P}_{new}$;
  until $|P_{new}|$ = *off_size*;
  $P' \leftarrow P \cup P_{new}$;
  Perform Mutation on each chromosome in P';
  Perform Local Improvement on each chromosome in P';
  $P \leftarrow \emptyset$;
  Put the *pop_size* best chromosomes from P' into P;
until max_iter *consecutive non-improving generations have occurred*;

In the remainder of this section, we describe the various components of our MA, namely the giant-tour chromosome representation, crossover operation, mutation operation and local improvement procedure. How the values for the various parameters involved in our approach are determined is discussed in Section 5.2.

4.1. Chromosome encoding and decoding

In our MA, each chromosome is a "giant-tour", which is simply a permutation of the set of nodes $\{1,\ldots,n\}$. This idea was first

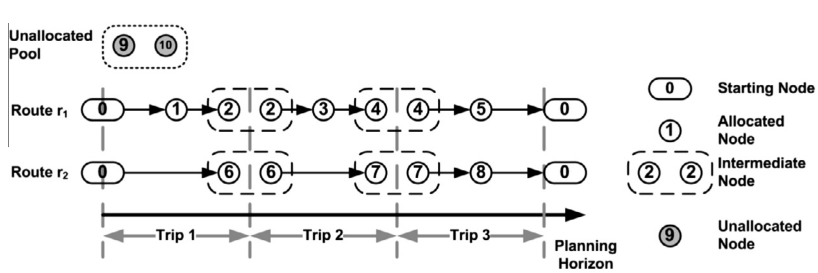


Fig. 1. An example of routes and trips.

proposed by Beasley (1983) as part of a route-first cluster-second heuristic, and was subsequently applied by Prins (2004) to the first competitive memetic algorithm for the VRP with chromosomes encoded as giant-tours and evaluated by a splitting procedure. Recently, this approach has been applied to other variants of the vehicle routing problem, including heterogeneous fleets (Prins, 2009), pickup and delivery VRP (Velasco et al., 2009) and a two-level (truck and trailer) routing problem (Villegas et al., 2010).

Given a giant-tour π, we can convert it into a solution to the mVRPP by partitioning it into $K+1$ sub-sequences. The first K sub-sequences correspond to the K vehicle routes, while the last sub-sequence contains the set of unvisited nodes. The solution is feasible if all of the K vehicle routes are feasible. For ease of discourse, we use the symbol S to represent a feasible partition of π as well as the mVRPP solution corresponding to that partition.

We now describe a greedy procedure called *split* that converts a given giant-tour $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$, $\pi_i \in \{1, \ldots, n\}$, into a feasible mVRPP solution S. Consider $\theta_k = (\pi_j, \pi_{j+1}, \ldots, \pi_n)$, where $k \leqslant K$, which is a sub-sequence of π. We can divide θ_k into two parts δ_k and θ_{k+1} by choosing a suitable cut-point i, $j \leqslant i \leqslant n$. The first part $\delta_k = (\pi_j, \ldots, \pi_i)$ is a sequence of nodes that can be converted into a feasible route r_k; the remaining nodes $\theta_{k+1} = (\pi_{i+1}, \ldots, \pi_n)$ are a new sequence of nodes. After K executions of this *split* procedure on π starting with $\theta_1 = \pi$, we have K feasible routes and a sequence θ_{K+1} of unvisited nodes. Note that if there does not exist an appropriate cut-point i, then the corresponding route will be empty.

For a given sequence θ_k, only some nodes can be a cut-point. Observe that for any node π_{i+1}, if $c_{0,\pi_{i+1}} > L$, then a feasible route cannot begin from π_{i+1}. Therefore, any node $\pi_i \in \theta_k$ can be classified into one of three sets V_A, V_B or V_C as follows:

- $\pi_i \in V_A$ if $(\pi_j, \ldots, \pi_i)$ can be converted into a feasible route and $c_{\pi_{i+1},0} \leqslant L$.
- $\pi_i \in V_B$ if $(\pi_j, \ldots, \pi_i)$ can be converted into a feasible route but $c_{\pi_{i+1},0} > L$.
- $\pi_i \in V_C$ if $(\pi_j, \ldots, \pi_i)$ cannot be converted into a feasible route.

Hence, i is not a valid cut-point if $\pi_i \in V_C$, but is a valid cut-point otherwise.

To determine if a sequence $(\pi_j, \ldots, \pi_i)$, $i \leqslant n$ can be converted into a feasible route, we first compute the vehicle's earliest arrival time at each node in θ_k. The earliest arrival time at node π_i is represented by a two-dimensional label:

$$l_i = (d_i, t_i),$$

where d_i, $1 \leqslant d_i \leqslant D$, denotes the arrival period and t_i represents the accumulated travel time since the beginning of period d_i. If a sequence $(\pi_j, \ldots, \pi_i)$, $i \leqslant n$ can be converted into a feasible route, then the label l_i satisfies one of the following conditions:

- $d_i = D$ and $t_i + c_{\pi_i,0} \leqslant L$;
- $d_i < D$ and $c_{\pi_i,0} \leqslant L$.

On the other hand, the sequence is infeasible if the label l_i satisfies one of the following cases:

Case 1 $d_i = D$ and $t_i + c_{\pi_i,0} > L$;
Case 2 $d_i > D$.

For Case 1, the node π_i can be reached in the last period, but the accumulated travel distance to the depot will exceed the limit. Since $c_{\pi_i,j} + c_{j,0} \geqslant c_{\pi_i,0}$ for all nodes j, all nodes after π_i will also not be valid cut-points. Case 2 indicates that the node cannot be reached within the planning horizon.

Algorithm 2. Greedy labelling procedure *getcut*

Input $\theta_k = (\pi_j, \pi_{j+1}, \ldots, \pi_n)$: a sequence of distinct nodes
Output i_{cut}: the index of the last node in route r_k
1 $i \leftarrow j$;
2 Current period $d \leftarrow 1$;
3 Accumulated travel time $t \leftarrow 0$;
4 Current node *last* $\leftarrow$ the depot;
5 Cut-point $i_{cut} \leftarrow$ the depot;
6 **while** $i \leqslant n$ **do**
7   **if** $t + c_{last,\pi_i} \leqslant L$ **then**
8     $t \leftarrow t + c_{last,\pi_i}$;
9     $last \leftarrow \pi_i$;
10   **if** $k < K$ **and** $((d_i = D$ **and** $t_i + c_{\pi_i,0} \leqslant L)$ **or** $(d_i < D$ **and** $c_{\pi_i,0} \leqslant L))$ **and** $c_{0,\pi_{i+1}} \leqslant L$ **then** $i_{cut} \leftarrow i$ $//\pi_i \in V_A$
11   **if** $k = K$ **and** $((d_i = D$ **and** $t_i + c_{\pi_i,0} \leqslant L)$ **or** $(d_i < D$ **and** $c_{\pi_i,0} \leqslant L))$ **then** $i_{cut} \leftarrow i$ $//\pi_i \in V_A \cup V_B$
12   **if** $d = D$ **and** $t + c_{\pi_i,0} > L$ **then break**;
13     $(d_i, t_i) \leftarrow (d, t)$;
14     $i \leftarrow i + 1$;
15   **else**
16     $d \leftarrow d + 1$;
17     $t \leftarrow 0$;
18   **If** $d > D$ **then break**;
19   **end**
20 **end**
21 **return** i_{cut};

Algorithm 3. The *split* procedure

Input $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$: the chromosome
Output $S = (\delta_1, \ldots, \delta_K, \delta_{K+1})$: an optimal partition of π
$k \leftarrow 1$;
$j \leftarrow 1$;
$\theta_1 \leftarrow (\pi_1, \ldots, \pi_n)$;
while $k \leqslant K$ **do**
  $i_{cut} \leftarrow getcut(\theta_k)$;
  $\delta_k \leftarrow (\pi_j, \ldots, \pi_{i_{cut}})$;
  if $k < K$ **and** δ_k *is empty* **then**
    $k \leftarrow K$;
  else
    $k \leftarrow k + 1$;
    $j \leftarrow i_{cut} + 1$;
  end
  $\theta_k \leftarrow (\pi_j, \ldots, \pi_n)$;
end

Given the above observations, we can use the greedy labelling procedure *getcut* given in Algorithm 2 to find the largest cut-point in a given sequence of distinct nodes $\theta_k = (\pi_j, \ldots, \pi_n)$. The procedure calculates the earliest arrival time l_i of the nodes sequentially. In each iteration, the current period d and the accumulated travel time t (where t cannot exceed L) is updated. As long as the current node is in V_A or V_B, it is a valid cut-point (lines 10–11). However, the labelling procedure terminates once the current node is in V_C, which is divided into Case 1 (line 12) and Case 2 (line 18). This procedure returns the largest valid cut-point, which maximizes the profit of the current route.

To convert a giant-tour chromosome into a feasible mVRPP solution, we use the procedure *split* shown in Algorithm 3 that employs *getcut* as a subroutine. The algorithm begins with the giant-tour π as the first sequence of distinct nodes θ_1. In each iteration, the maximal cut-point i_{cut} is determined by the *getcut* procedure, and the route $\delta_k = (\pi_j, \ldots, \pi_{i_{cut}})$ and a new sequence of distinct nodes θ_{k+1} are generated. Note that if δ_k is empty, then we simply label the last sequence δ_K (line 7).

Theorem 1. *For the given giant-tour π, the partition S found by the split procedure is optimal in terms of collected reward.*

Proof. Let S^* be an optimal partition of π, and assume sequence $\delta_h^* = (\pi_j, \ldots, \pi_i)$ is the first sequence in S^* that is different from $S = (\delta_1, \ldots, \delta_K, \delta_{K+1})$.

When $h = K$, π_i must be the node with the largest index in the node set $V_A \cup V_B$ of δ_K. However, this is the node chosen by the *split* procedure, which contradicts the assumption that δ_h^* is the first sequence that is different from S.

When $h < K$, the corresponding sequence in S is $\delta_h = (\pi_j, \ldots, \pi_{i'})$ and $i' > i$. We have $\delta_h^* = (\pi_j, \ldots, \pi_i)$ and $\delta_{h+1}^* = (\pi_{i+1}, \ldots, \pi_l)$. Observe that when we replace δ_h^* and δ_{h+1}^* with $(\pi_j, \ldots, \pi_{i+1})$ and $(\pi_{i+2}, \ldots, \pi_l)$ (i.e., set the first node in δ_{h+1}^* to be the last node in δ_h^*), the resultant solution is still feasible and the same amount of reward is collected. We can repeat this procedure until the resultant solution is the same as S. Therefore, S is optimal. $\square$

The following example demonstrates our *split* procedure. Suppose the input graph G is as given by Fig. 2a, the number of vehicles $K = 2$, the number of periods $D = 3$, the working time limit $L = 20$, and each node i, $i = 1, \ldots, n$ has a service time $s_i = 2$. The corresponding bidirected graph G' for the alternative problem representation is given by Fig. 2b. Consider the giant-tour $\pi = (1,2,3,4,5,6,7,8,9,10)$; the solution produced by the *split* procedure is given by Table 1a, which obtains the rewards for nodes 1–8. In contrast, a naive greedy procedure that simply maximizes the number of nodes visited per route would produce the solution given by Table 1b, which only visits nodes 1–5 because the cost of the edge from the depot to node 6 exceeds L. This example illustrates why the consideration of nodes in the sets V_A and V_B is required to find the optimal partition.

We now show that the *split* procedure runs in $O(n)$ time. Given an optimal partition $S = (\delta_1, \delta_2, \ldots, \delta_K, \delta_{K+1})$ generated by the *split* procedure, consider $\delta_k = (\pi_j, \ldots, \pi_i)$ and $\delta_{k+1} = (\pi_{i+1}, \ldots, \pi_h)$, $1 \leqslant k < K$. Observe that the *getcut* subroutine, when given θ_k as input, labels fewer than $|\delta_k| + |\delta_{k+1}|$ nodes; if this is not the case, then π_h will be labelled with the label l_h and does not fulfil either termination criterion (i.e., $d_h \leqslant D$ and $t_h + c_{\pi_h,0} \leqslant L$). This implies that $(\pi_j, \ldots, \pi_h)$ can be converted into a feasible route, which contradicts the fact that *getcut* returns the largest cut-point. Since the total number of nodes labelled is less than $\sum_{k=1}^{K} |\delta_k| + |\delta_{k+1}| < 2n$, the *split* procedure runs in $O(n)$ time.

Table 1
Solutions produced on the example problem.

(a) Solution by split procedure		
Route 1		Length
Trip 1	$0 \to 1 \to 2$	16
Trip 2	$2 \to 3 \to 4$	19
Trip 3	$4 \to 0$	13
Route 2		Length
Trip 1	$0 \to 5$	17
Trip 2	$5 \to 6 \to 7$	19
Trip 3	$7 \to 8 \to 0$	20
(b) Solution by naive greedy procedure		
Route 1		Length
Trip 1	$0 \to 1 \to 2$	16
Trip 2	$2 \to 3 \to 4$	19
Trip 3	$4 \to 5 \to 0$	20
Route 2		Length
Trip 1	$0 \to 0$	0
Trip 2	$0 \to 0$	0
Trip 3	$0 \to 0$	0

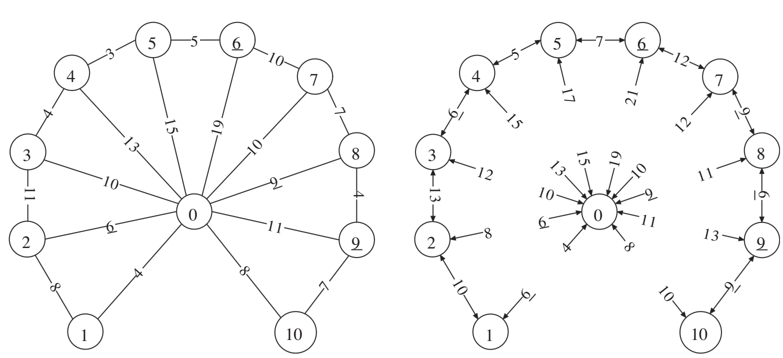
Note that we now have an $O(n \cdot n!)$ exact algorithm for the mVRPP by trying each of the $n!$ possible giant-tours (which is just a permutation of $\{1, \ldots, n\}$) and finding its optimal partition using *split* in $O(n)$ time.

4.2. Crossover

In each generation, we use an *order-based crossover* (OX) operator to generate all offspring chromosomes, which is a common crossover operator for both the travelling salesman problem (TSP) (Oliver et al., 1987) and the VRP (Prins, 2004). We performed some preliminary experiments with both the one-point and the two-point versions of the OX operator; the results suggested that both operators provide similar performance. Hence, we selected the simpler one-point crossover operator for our approach.

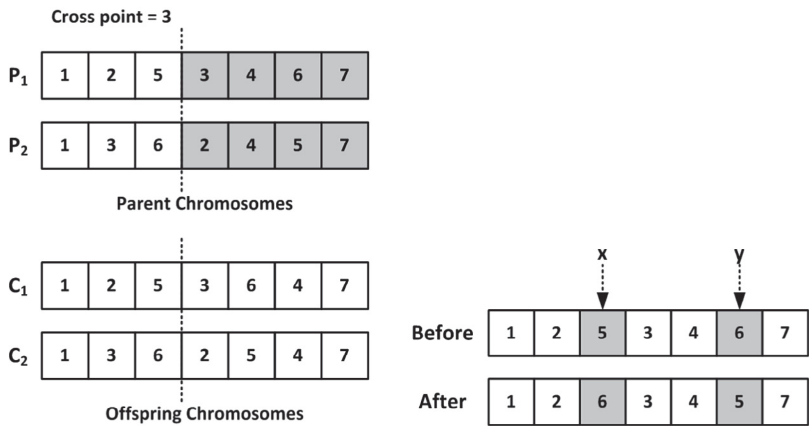
Given two randomly selected parent chromosomes, the one-point OX operation consists of three steps. First, a crossing point (between two genes) is selected at random. Next, the nodes of each chromosome up to the crossing point are copied to the offspring chromosomes C_1 and C_2, respectively, at the same locations. Finally, the remaining genes of each offspring are constructed in the order determined by the other parent chromosome.

Fig. 3a shows an example of the one-point OX operation on two randomly selected parent chromosomes P_1 and P_2 where the selected crossing point is between positions 2 and position 3. The offspring chromosome C_1 is generated by first copying the left sub-sequence of P_1, i.e. (1,2,5). We then scan each gene in P_2 in order, and copy the non-duplicated genes to the end of C_1 in that



(a) Graph G (b) Graph G'

Fig. 2. An mVRPP example.



(a) An example of one-point crossover

(b) An example of mutation operator

Fig. 3. Examples of memetic operators.

order. The other offspring chromosome C_2 is obtained in the same way.

4.3. Mutation

After applying the crossover procedure, our candidate pool consists of both the parent and the offspring population. We then perform a mutation operation on every chromosome, which introduces some random variation into the chromosomes. The aim is to prevent the algorithm from being trapped in local optima and to maintain diversity in the population.

Our mutation operation works as follows. Each gene in the chromosome has a probability of ρ to be chosen for mutation. If chosen, the gene is swapped with another randomly selected gene. Fig. 3b shows an example where a single gene was selected for mutation. We empirically determined that $\rho = 0.005$ is a suitable value for our test data (see Section 5.2).

4.4. Local improvement process

After the crossover and mutation operations, we apply a local improvement approach to improve chromosomes in the candidate pool. Let $W(S)$ and $t(S)$ be the total reward and total travel time for a solution S, respectively. Algorithm 4 shows the overall process of our approach, which is applied on a single giant-tour chromosome.

Algorithm 4. Local improvement procedure

$S \leftarrow$ input chromosome;
repeat
  for each operator $op \in \{exchange, 2-opt, relocate, segment-move\}$ **do**
    $S' \leftarrow op(S)$;
    if $W(S') > W(S)$ **or** $(W(S') = W(S)$ **and** $(t(S') < t(S))$ **then** $S \leftarrow S'$;
  then
until LIP_iters *consecutive non-improving iterations have occurred*;
$S \leftarrow TSPP(S)$;

In each iteration, we first apply four simple heuristic operators *exchange, 2-opt, relocate* and *segment-move* on the chromosome in this order. Given the current chromosome $S = (\pi_1, \pi_2, \ldots, \pi_n)$, let x, y and $z, x < y < z$, be three uniformly randomly selected distinct indices in the chromosome. These operators modify it to produce a new chromosome S' as follows:

Exchange Exchange π_x and π_y, i.e., $S' = (\pi_1, \ldots, \pi_{x-1}, \pi_y, \pi_{x+1}, \ldots, \pi_{y-1}, \pi_x, \pi_{y+1}, \ldots, \pi_n)$; this is similar to our mutation operator.
2-opt Reverse the sequence $(\pi_x, \ldots, \pi_y)$ in place, i.e., $S' = (\pi_1, \ldots, \pi_{x-1}, \pi_y, \pi_{y-1}, \ldots, \pi_x, \ldots, \pi_n)$.
Relocate Move π_y to the position just before π_x, i.e. $S' = (\pi_1, \ldots, \pi_y, \pi_x, \ldots, \pi_{y-1}, \pi_{y+1}, \ldots, \pi_n)$.
Segment-move Swap two consecutive segments $(\pi_x, \ldots, \pi_y)$ and $(\pi_{y+1}, \ldots, \pi_z)$, i.e., $S' = (\pi_1, \ldots, \pi_{x-1}, \pi_{y+1}, \ldots, \pi_z, \pi_x, \ldots, \pi_y, \pi_{z+1}, \ldots, \pi_n)$.

Given the new solution S' generated by an operator and the original current solution S, the new solution replaces the current solution if its total reward is greater, or if its reward is equal but total travel time is less. We repeat this process until *LIP_iters* consecutive iterations have occurred where the current solution S has not been replaced. In our implementation, we set *LIP_iters* = 800 after some preliminary experiments (see Section 5.2).

Finally, we attempt to improve the resultant solution using a dynamic programming (DP) approach. Given the nodes for a trip within a period, finding the path with the lowest travel cost for these nodes is a travelling salesman path problem (TSPP), which is a well-known NP-hard problem. However, when the number of nodes visited is small (e.g., $\leqslant$10 nodes), the dynamic programming algorithm proposed by Bellman (1962) is able to find the optimal sequence efficiently. Hence, for each sub-sequence $r_d^k = (\pi_x, \ldots, \pi_y)$ representing a trip from S, we perform this DP algorithm on the set of nodes $\{\pi_{x+1}, \ldots, \pi_{y-1}\}$, and replace the sub-sequence with the solution. Note that the starting node π_x and the ending node π_y of the trip are considered to be the fixed source and destination, respectively. Moreover, since the DP always generates a sequence with travel cost no more than the original sequence, we always perform the replacement.

5. Experiments and analysis

In order to evaluate the performance of our proposed MA, we conducted several computational experiments on two classes of test data. The algorithm was coded in C++, and all experiments were performed on a PC with a 2.27 gigahertz Xeon processor and 8 gigabyte of RAM.

5.1. Computational data

Our computational data is divided into two classes. The **Class I** instances were derived from existing benchmark data for the team orienteering problem (TOP), which is identical to the mVRPP except that it only involves a single period. This data was originally generated by Chao (1993), and was subsequently filtered by Archetti et al. (2007) who removed the instances that had obvious optimal solutions or were obviously infeasible. Although there are seven sets of data, we only consider set 7, which is the only group where the starting and ending destinations are identical. The graph G for all instances in this class are identical (taken from an instance of the period routing problem by Christofides and Beasley (1984)), which consists of 102 nodes with a total profit of 1498; the instances differ only in the maximum travel time T_{max} and the number of vehicles $K = 2,3,4$. Currently, the best TOP results are held by Bouly et al. (2010) and Vansteenwegen et al. (2011).

Given a TOP instance with maximum travel time T_{max}, we can convert it into an mVRPP instance by dividing the single working period into D sub-periods, where the working time restriction for each period is set to be $L = T_{max}/D$. We modified the 43 instances of the original test set in this manner for $D = 2$ and $D = 4$. Combined with the unmodified data ($D = 1$), we have a total of 129 instances in Class I. All other characteristics (e.g., the location coordinates, node rewards and the number of vehicles) were unchanged, and service times are ignored, i.e., $\forall_{i \in V} s_i = 0$. The original TOP instances can be found at http://prolog.univie.ac.at/research/OP/.

The **Class II** instances were derived from the 8 benchmark instances for the vehicle routing problem with distance restriction (DVRP) created by Golden et al. (1998). The DVRP differs from the mVRPP in that the objective is to fulfil the demand at all locations using the lowest travel cost given a distance restriction T_{max}, and there is only one period. The customers in these instances are located in concentric circles around the depot. To the best of our knowledge, the latest investigation of the DVRP on this data set with sufficiently detailed results for comparison was conducted by Groër (2008); while there are more recent publications on the DVRP (Nagata and Bräysy, 2009; Prins, 2009), they do not provide individual route details.

We convert a given DVRP instance with maximum travel time T_{max} into an mVRPP instance as follows. Given the best solution found by Groër (2008) for the DVRP, we set the number of routes K to be the number of vehicles used in the solution, and set T_{max} to be the longest time taken by any vehicle in the solution. Then, we set the total working time restriction per period to be $L = T_{max}/D$. We performed this modification for $D = 1,2,4$. The demand for each node is assigned in two ways:

- For the Group 1 instances, the reward for each node $w_i = 1$.
- For the Group 2 instances, the reward w_i for each node is the demand in the original instances by Golden et al. (1998).

Group 1 produces mVRPP instances in which the goal is to cover as many nodes as possible, while Group 2 simply follows the demand distribution by Golden et al. (1998). There are 24 instances per group (eight instances for each of $D = 1,2,4$), for a total of 48 Class II instances. Once again, the service times for all nodes are set to zero. Note that these instances contain 240–440 nodes and the number of vehicles K ranges from 5 to 10, so the scale of these instances is larger than for Class I. To download the Class II test data, it can be found at http://www.rhsmith.umd.edu/faculty/bgolden/vrp_data.htm.

5.2. Parameter tuning

Our parameter tuning tests were performed using six of the Class I instances where $D = 1$, namely *p7.2.s*, *p7.2.t*, *p7.3.s*, *p7.3.t*, *p7.4.s* and *p7.4.t*. For each instance, we conducted five independent runs using different random seeds and took the best solution. Because the MA approach contains five parameters, in order to find appropriate values for these parameters, we conducted two parameter tuning experiments.

In the first experiment, we deactivated the mutation operation by setting the mutation probability $\rho = 0$, and then used a full factorial design to determine the remaining four parameters. The maximum number of consecutive non-improving generations *max_iter* $\in \{50,100\}$ determines the amount of time invested to search for the optimal solution. The number of iterations of neighbourhood operations *LIP_iters* $\in \{200,500,800\}$ in the local improvement process controls the intensification of the search. Finally, both the size of the retained population *pop_size* $\in \{5,10,20\}$ and the number of offspring generated *off_size* $\in \{5,10,20\}$ affect the convergence reliability and speed of the overall algorithm. Due to page limit, the table of detailed results is not reported in this paper. The experiment shows that using the values *max_iter* = 100, *LIP_iter* = 800, *pop_size* = 20, and *off_size* = 20 produces the best solutions using just under 22 CPU seconds on average, which is sufficiently fast for our purposes. Consequently, we adopt these values for the remainder of our experiments.

Furthermore, the experiment reveals that increasing any of the three parameters *max_iter*, *LIP_iter* or *pop_size* produces an improvement in solution quality at the expense of added computation time. Therefore, the practitioner can further tune the algorithm in order to balance the trade-off between solution quality and computational time depending on the practical situation. Table 2 shows the percentage improvement in the solution quality of our MA approach when we double one of these three parameters, along with the percentage increase in computation time. We see that doubling the number of non-improving iterations *LIP_iter* of the local improvement procedure has the largest positive impact on solution quality, but requires almost double the computation time. Doubling the overall number of non-improving iterations *max_iter* or the population size *pop_size* produces more modest improvements and uses correspondingly less time. Finally, the last row shows that omitting the TSPP dynamic program in the local improvement procedure would speed up the algorithm by about 15% for a slight decrease in solution quality.

The purpose of the second experiment is to determine an appropriate mutation probability ρ. Using the above parameters, we

Table 2
Trade-off between solution quality and computation time.

Modification	Improvement (%)	Time increment (%)
Set *max_iter* = 200, *LIP_iter* = 800, *pop_size* = 20	0.302	77.5
Set *max_iter* = 100, *LIP_iter* = 1600, *pop_size* = 20	0.363	95.6
Set *max_iter* = 100, *LIP_iter* = 800, *pop_size* = 40	0.272	67.1
Disable TSPP operator	−0.055	−14.7

Table 3
Parameter tuning for ρ.

max_iter	*LIP_iter*	*pop_size*	*off_size*	ρ	Avg. gap (%)	Avg. CPU
100	800	20	20	0	1.14	21.97
100	**800**	**20**	**20**	**0.005**	**0.88**	**38.47**
100	800	20	20	0.01	1.17	38.07

tested the values of $\rho = \{0, 0.005, 0.01\}$. As the results in Table 3 show, the value of $\rho = 0.005$ produced the best performance. Hence, we set $\rho = 0.005$ for the remainder of our experiments.

5.3. Results for Class I instances

We compared the performance of our MA approach on the Class I instances with $D = 1$ with the best existing TOP approaches in literature. These algorithms are:

- *TMH*: A tabu search heuristic by Tang and Miller-Hooks (2005)
- *GTP*: A tabu search with penalty strategy by Archetti et al. (2007)
- *ASe*: A sequential ant colony optimization technique by Ke et al. (2008)
- *FPR*: The fast path relinking approach by Souffriau et al. (2010)
- *SPR*: The slow path relinking approach by Souffriau et al. (2010)

We also devised an iterated local search (ILS) approach that is similar to our MA approach except for the following changes: (1) size of the population *pop_size* is one; (2) the crossover operation is omitted; and (3) the mutation probability ρ is set to 0.05 so that the mutation operator works as the perturbation device for the iterated local search. We use this ILS approach primarily to provide a basis of comparison for our MA for the instances with $D = 2,4$.

In addition, we also provide a comparison with five simple heuristics that a human scheduler would employ. The five heuristics are:

H1 Begin with an empty route r_1. Repeatedly append the feasible node resulting in the smallest added travel distance to r_1 until no such node exists. Continue this process with $r_2, \ldots, r_K$.
H2 Same as H1, except the node appended is the one with smallest travel distance divided by reward.
H3 Begin with K empty routes $r_1, \ldots, r_K$. Set $d = 1$. Repeatedly append the feasible node resulting in the smallest added travel distance to period d of r_1 until no such node exists. Perform this process with $r_2, \ldots, r_K$. Continue with $d = 2$, starting once again with r_1, until the trips for $d = D$ periods for all routes have been constructed.
H4 Same as H3, except the node appended is the one with smallest travel distance divided by reward.
H5 Begin with K empty routes $r_1, \ldots, r_K$. Select the feasible node with the largest reward, and append it to the route resulting in the smallest added travel distance. Repeat until no such node exists.

Heuristics H1 and H2 construct the solution one route at a time, heuristics H3 and H4 perform the construction one period at a time, and heuristic H5 greedily adds the node with the greatest reward into the solution. We have conferred with the planners, who confirmed that their human-generated schedules were produced using similar heuristic rules.

The results are summarized in Table 4. The meanings of *Avg Gap* and *Avg CPU* are the same as for Table 3. In terms of average gap, our MA approach ranks third with a gap of 0.071%, which is less than 0.1% behind ASe (0.002%) and SPR (0.041%). This result is achieved in about 3 min, which is a shortter running time compared to ASe and SPR on a comparable machine configuration. Our approach is therefore competitive when applied to the TOP even though it is designed for the mVRPP (the existing TOP approaches cannot be applied to the mVRPP without major modification). Note that the gap between the best greedy heuristic and the best known TOP solutions is above 22% on average, which indicates that the greedy approaches that a human scheduler would use are relatively poor.

Table 5 summarizes the results for all the Class I instances with $D = 1,2,4$, where we compare the performance of our MA and ILS algorithms. The column *#instances* gives the number of instances in the test set, the column K is the number of vehicles, and the column *Greedy Gap* gives the gap between the best solutions found by the five greedy heuristics and the best known TOP solutions.

For the multiperiod instances ($D = 2$ and $D = 4$), there are no existing results to provide a completely fair comparison. However, the best known solution for the corresponding single period ($D = 1$) instance is likely to be an upper bound. Thus, we use of the same best known solutions for the single period instances as a reference to evaluate the quality of our approach, although naturally the gaps will be larger.

Our experiments confirmed that for multiperiod instances, the number of assigned tasks is less than the corresponding single

Table 4
Comparison with TOP approaches on Class I instances with $D = 1$.

	TMH	FPR	GTP	SPR	ASe	MA	ILS	Greedy
Avg. gap (%)	1.154	0.536	0.429	0.041	0.002	0.071	1.930	22.10
Avg. CPU	432.6	6.3	158.97	272.8	303.5	182.6	7.0	–

Table 5
Summarized results for MA on Class I instances.

Periods	#instances	K	Greedy gap (%)	MA		ILS	
				Avg. gap (%)	Avg. CPU	Avg. gap (%)	Avg. CPU
$D = 1$	17	2	22.70	0.04	14.27	2.025	0.47
	13	3	19.63	0.13	18.51	2.034	0.79
	13	4	23.79	0.06	23.22	1.702	0.91
Average	–	–	22.10	0.07	18.26	1.930	0.70
$D = 2$	17	2	25.17	1.69	15.51	4.009	0.51
	13	3	22.25	1.95	24.74	4.578	0.86
	13	4	26.71	4.02	24.91	5.977	0.99
Average	–	–	24.75	2.47	21.14	4.776	0.76
$D = 4$	17	2	40.08	7.41	15.82	9.594	0.53
	13	3	43.99	7.88	22.94	10.830	0.78
	13	4	53.68	21.20	24.44	23.152	0.70
Average	–	–	45.38	11.72	20.57	14.067	0.66

period instance and results in a decrease in total reward obtained. This shows that the number of periods is a significant factor. Hence, approaches that do not explicitly consider the effect of working hour restrictions may not be directly portable to problems where the planning horizon is periodic.

The detailed experimental results for the Class I instances are reported in Table 6, and can serve as benchmarks for future researchers on this problem. Interestingly, the solution produced by our MA approach for the instance *p7.3.t* improved on the best known solution.

5.4. Results for Class II Instances

Recall that the Class II instances are derived from DVRP instances based on the solutions found by Groër (2008). When $D = 1$, the route corresponding to this solution (that visits all nodes) is optimal for the mVRPP instance, with a total reward of $\sum_{i \in V} w_i$. When $D = 2$ or $D = 4$, we use the same total reward value $\sum_{i \in V} w_i$ as a naive upper bound on the solution for comparison.

Our experimental results on Group 1 of the Class II instances, where the reward for each node is 1, is reported in Table 7. For each instance, we report the worst reward (Z_{min}), the best reward (Z_{max}), the average reward (Z_{avg}) and the average travel distance (C_{avg}) from 10 independent runs, along with the average CPU time required by the corresponding algorithm to find its final solution (*Avg CPU (s)*). The column *G_Gap* gives the average gap from the best of the five greedy heuristics; this gives an indication of the amount of improvement our MA approach obtains compared to a human scheduler who employs simple rules of thumb. We also report under column *Gap* the average gap between the obtained solutions and the total profit of all nodes as given by *TP*.

For $D = 1$, the average gap between the solutions generated by MA and the upper bound (which is optimal in this case) is 2.58%. Considering the scale of the problem, these results show that the proposed MA approach can effectively find solutions that are close to optimal in a reasonable amount of time (just over 6 minute on average). Our ILS approach can also find reasonable solutions (with an average gap from optimal of 5.22%) using less than 7 second on average, so our ILS approach is a reasonable alternative if time is a crucial limiting factor. The quality of the solutions is also highlighted by the fact that the best of the 5 greedy heuristics can only find solutions that are 10.68% away from optimal on average.

When $D = 2$ and $D = 4$, the average gaps from the naive upper bound for MA are 2.79% and 4.92%, respectively. For the solutions generated by ILS, the average gaps are 7.10% and 8.43%, respectively. Unfortunately, it is difficult to judge the performance of

Table 6
Detailed results for MA on Class I instances.

Instance	Best known	$D = 1$			$D = 2$			$D = 4$		
		Z_{max}	Z_{avg}	Avg. CPU	Z_{max}	Z_{avg}	Avg. CPU	Z_{max}	Z_{avg}	Avg. CPU
p7.2.d	190	190	190	0.383	179	179	0.235	105	105	1.301
p7.2.e	290	290	289.6	1.251	276	274.9	2.246	254	254	0.72
p7.2.f	387	387	386.3	3.328	377	377	2.989	333	331	3.954
p7.2.g	459	459	459	3.775	455	451.6	5.347	428	423.6	7.622
p7.2.h	521	521	521	4.324	517	517	3.398	498	498	3.366
p7.2.i	580	580	577.9	13.289	575	572.6	7.872	560	558	7.206
p7.2.j	646	646	640.7	12.164	639	635.8	19.064	615	610.8	16.649
p7.2.k	705	705	700.2	9.008	697	691.8	10.426	679	675.4	11.363
p7.2.l	767	767	761.2	14.442	760	753.4	21.146	733	727.2	11.133
p7.2.m	827	827	820.5	20.083	817	808.2	19.983	788	783	11.966
p7.2.n	888	887	873	15.549	871	856.8	11.96	842	835.1	20.757
p7.2.o	945	945	936.6	16.419	929	920.3	19.078	914	902	24.255
p7.2.p	1002	1002	989.6	26.009	986	974.5	17.202	959	954.7	30.421
p7.2.q	1044	1044	1039.3	22.527	1040	1033.3	29.007	1018	1006.4	32.193
p7.2.r	1094	1093	1086.8	28.153	1084	1077.8	33.95	1059	1053.3	21.507
p7.2.s	1136	1131	1123.2	22.83	1128	1120.5	31.934	1106	1097.4	29.951
p7.2.t	1179	1179	1169.5	29.084	1162	1151.9	27.85	1161	1151.7	34.522
p7.3.h	425	425	424.3	9.167	417	417	1.173	332	332	1.575
p7.3.i	487	487	485.5	5.139	474	473.9	9.823	427	425	6.654
p7.3.j	564	564	561.5	9.534	555	551	20.396	500	496.9	15.75
p7.3.k	633	633	632	12.828	624	615.4	14.426	589	580.4	4.572
p7.3.l	684	684	682	9.859	675	673.5	12.416	629	628.3	10.051
p7.3.m	762	762	753.1	13.442	734	726.1	15.322	704	690.9	20.35
p7.3.n	820	820	816.6	24.269	800	798.4	20.987	757	755.8	24.692
p7.3.o	874	874	873.9	23.15	853	849.6	25.211	818	807	32.027
p7.3.p	929	925	920.6	21.193	912	904.6	37.702	899	897.1	24.95
p7.3.q	987	987	981.8	30.883	969	962.8	32.934	948	941.3	26.981
p7.3.r	1026	1014	1009.7	22.894	1011	1003.3	44.053	984	977.8	22.848
p7.3.s	1081	1078	1063.3	23.543	1052	1042.6	41.534	1031	1022.8	56.418
p7.3.t	1118	**1120**	1110.7	34.778	1116	1098.1	45.692	1070	1063.7	51.3
p7.4.g	217	217	217	0.718	189	189	0.895	63	63	0.235
p7.4.h	285	285	285	4.694	269	269	0.624	111	111	0.251
p7.4.i	366	366	366	2.28	342	342	0.991	267	267	0.814
p7.4.k	520	520	518.4	17.807	507	505.4	11.332	383	383	7.28
p7.4.l	590	590	588.8	19.319	568	568	9.074	498	498	10.798
p7.4.m	646	646	645.2	16.461	641	636.8	11.144	563	561.6	14.128
p7.4.n	730	727	725.7	15.074	701	698.6	37.419	627	619.5	19.499
p7.4.o	781	781	778.8	44.992	754	746.7	20.964	726	726	17.59
p7.4.p	846	846	839.4	21.549	822	820.9	42.665	763	756.6	23.629
p7.4.q	909	907	903.1	18.629	882	879.5	44.205	835	823.8	54.024
p7.4.r	970	970	964.6	36.024	943	936.7	33.058	889	880.8	46.482
p7.4.s	1022	1021	1016.6	50.715	1004	996.6	56.672	955	947.9	42.683
p7.4.t	1077	1077	1076.3	53.629	1051	1041.1	54.82	994	988.7	80.248

Table 7
Results on Group 1 of the Class II instances.

Instance	D	K	L	TP	G_Gap (%)	MA						ILS					
						Z_{max}	Z_{min}	Z_{avg}	C_{avg}	Avg. CPU	Gap (%)	Z_{max}	Z_{min}	Z_{avg}	C_{avg}	Avg. CPU	Gap (%)
G.01	1	9	648	240	6.25	237	234	235.2	5671.72	130.972	1.25	234	223	227.8	5669.9	3.52	2.50
G.02	1	10	900	320	7.19	317	312	314.4	8689.97	313.441	0.94	312	297	299.9	8779.2	2.69	2.50
G.03	1	10	1171	400	12.00	389	381	385.4	11,401	480.416	2.75	380	362	369.6	11330.8	9.22	5.00
G.04	1	10	1410	480	11.25	461	452	456.7	13935.2	784.98	3.96	444	436	440.8	13821.8	15.66	7.50
G.05	1	5	1302	200	10.00	195	184	188.1	6449.7	74.264	2.50	186	180	183.8	6413.7	1.37	7.00
G.06	1	7	1222	280	16.79	266	260	263	8425.44	160.165	5.00	267	253	258.6	8380.5	7.93	4.64
G.07	1	9	1186	360	11.94	353	341	347	10498.2	383.312	1.94	344	329	333.3	10403.4	10.04	4.44
G.08	1	10	1200	440	10.00	430	416	423.6	11774.6	685.652	2.27	404	397	400	11715.2	5.36	8.18
Avg.	–		–	–	10.68	–	–	–	–	376.65	2.58	–	–	–	–	6.97	5.22
G.01	2	9	324	240	7.08	237	229	232.1	5597.6	155.373	1.25	228	224	225.3	5667.0	0.84	5.00
G.02	2	10	450	320	9.38	317	311	313.7	8659.64	343.151	0.94	306	294	300.6	8612.0	8.56	4.38
G.03	2	10	585.5	400	13.25	390	379	381.7	11308.6	546.88	2.50	370	355	363.5	11102.5	10.82	7.50
G.04	2	10	705	480	11.46	457	435	450.2	13762.1	860.666	4.79	442	432	435.9	13704.3	13.78	7.92
G.05	2	5	651	200	10.00	196	179	187.7	6383.56	74.674	2.00	186	180	181.1	6387.8	0.68	7.00
G.06	2	7	611	280	16.79	266	258	261.3	8364.07	203.335	5.00	257	242	251.1	8231.4	6.26	8.21
G.07	2	9	593	360	12.50	353	337	346.3	10399.9	402.487	1.94	334	325	329.6	10188.6	8.10	7.22
G.08	2	10	600	440	11.36	423	413	418	11679.4	698.767	3.86	398	394	395.4	11574.2	7.20	9.55
Avg.	–		–	–	11.48	–	–	–	–	410.667	2.79	–	–	–	–	7.03	7.10
G.01	4	9	162	240	16.67	233	220	227.8	5501.82	159.381	2.92	225	215	219.2	5368.5	3.82	6.25
G.02	4	10	225	320	10.94	313	305	309.2	8479.2	380.191	2.19	298	285	289	8294.9	4.04	6.88
G.03	4	10	292.75	400	16.25	386	373	378.2	11084.5	768.215	3.50	369	348	358.8	10926.0	14.20	7.75
G.04	4	10	352.5	480	14.17	448	437	443.3	13566.9	1080.86	6.67	430	420	424.1	13376.2	15.87	10.42
G.05	4	5	325.5	200	12.00	182	180	181	6241.98	72.449	9.00	181	177	178.9	6166.3	1.79	9.50
G.06	4	7	305.5	280	17.86	263	254	257	8178.26	200.391	6.07	256	243	248.7	8050.8	6.38	8.57
G.07	4	9	296.5	360	12.50	344	336	340.9	10184.9	485.769	4.44	330	323	326.5	10149.2	8.01	8.33
G.08	4	10	300	440	14.55	420	411	416.7	11528.3	830.589	4.55	397	386	389.7	11353.4	8.01	9.77
Avg.	–		–	–	14.37	–	–	–	–	497.231	4.92	–	–	–	–	7.77	8.43

these approaches based on these values since the upper bounds are not known to be tight. On the other hand, the average gaps for the greedy approaches is 11.48% when $D = 2$ and 14.37% when $D = 4$, respectively, which shows that our approaches significantly outperform human schedulers.

The results for Group 2 are reported in Table 8. For $D = 1$, the average gap from the upper bound for MA is 0.96% while it is 2.40% for ILS. This shows that the proposed approaches can efficiently find solutions that are very close to optimal. When $D = 2$ and $D = 4$, the average gaps from the upper bound for MA are

Table 8
Results on Group 2 of the Class II instances.

Instance	D	K	L	TP	G_Gap (%)	MA						ILS					
						Z_{max}	Z_{min}	Z_{avg}	C_{avg}	Avg. CPU	Gap (%)	Z_{max}	Z_{min}	Z_{avg}	C_{avg}	Avg. CPU	Gap (%)
G.01	1	9	648	4800	4.17	4780	4710	4750	5713.7	176.52	0.42	4730	4660	4692	5678.96	3.52	1.46
G.02	1	10	900	6400	7.19	6390	6330	6368	8745.7	346.53	0.16	6350	6140	6249	8730.45	9.05	0.78
G.03	1	10	1171	8000	6.00	7950	7710	7844	11403.2	584.80	0.63	7810	7680	7757	11254.2	13.76	2.38
G.04	1	10	1410	9600	8.13	9400	9160	9267	13879.2	1002.92	2.08	9310	9160	9209	13851.6	20.94	3.02
G.05	1	5	1302	4000	10.00	3990	3830	3897	6457.5	86.79	0.25	3860	3730	3815	6392.13	3.13	3.50
G.06	1	7	1222	5600	9.29	5500	5370	5443	8444.1	211.38	1.79	5460	5360	5421	8360.28	5.84	2.50
G.07	1	9	1186	7200	6.53	7110	6990	7066	10458.8	495.89	1.25	7010	6950	6980	10282.5	10.85	2.64
G.08	1	10	1200	8800	6.48	8700	8530	8601	11821.8	767.57	1.14	8540	8450	8506	11786.1	14.53	2.95
Avg.	–		–	–	7.22	–	–	–	–	459.05	0.96	–	–	–	–	10.20	2.40
G.01	2	9	324	4800	4.79	4740	4680	4715	5638.9	177.57	1.25	4690	4640	4664	5595.08	3.91	2.29
G.02	2	10	450	6400	7.97	6390	6290	6337	8634.8	441.04	0.16	6300	6190	6250	8628.32	11.70	1.56
G.03	2	10	585.5	8000	7.63	7910	7780	7845	11346.3	715.87	1.13	7860	7640	7738	11218.3	14.22	1.75
G.04	2	10	705	9600	7.71	9370	9050	9266	13780.9	1119.51	2.40	8990	8780	8922	13592.8	21.05	6.35
G.05	2	5	651	4000	9.00	3970	3820	3866	6370.7	88.47	0.75	3830	3710	3764	6271.84	3.13	4.25
G.06	2	7	611	5600	9.29	5480	5400	5434	8383.5	259.25	2.14	5480	5350	5395	8311.12	6.07	2.14
G.07	2	9	593	7200	6.53	7080	6980	7034	10422.5	574.03	1.67	6970	6870	6931	10,242	9.18	3.19
G.08	2	10	600	8800	8.07	8580	8360	8500	11656.8	800.47	2.50	8520	8370	8446	11652.7	18.71	3.18
Avg.	–		–	–	7.62	–	–	–	–	522.03	1.36	–	–	–	–	11.00	3.09
G.01	4	9	162	4800	18.33	4700	4550	4636	5463.1	179.99	2.08	4570	4450	4497	5410.44	5.09	4.79
G.02	4	10	225	6400	11.41	6300	6230	6266	8488.4	452.96	1.56	6250	6030	6099	8352.77	9.55	2.34
G.03	4	10	292.75	8000	23.50	7840	7580	7737	11132.8	765.48	2.00	7660	7530	7606	10959.9	13.27	4.25
G.04	4	10	352.5	9600	15.21	9220	8890	9081	13533.8	1053.80	3.96	8990	8770	8816	13221.6	21.62	6.35
G.05	4	5	325.5	4000	8.25	3850	3760	3803	6247.2	109.43	3.75	3760	3620	3683	6075.75	3.19	6.00
G.06	4	7	305.5	5600	9.64	5390	5300	5342	8200.9	255.53	3.75	5360	5200	5322	8053.62	6.17	4.29
G.07	4	9	296.5	7200	11.25	7020	6800	6963	10174.2	535.10	2.50	6950	6810	6872	10138.9	10.32	3.47
G.08	4	10	300	8800	14.09	8590	8370	8473	11473.3	931.52	2.39	8400	8260	8345	11391.7	18.03	4.55
Avg.	–		–	–	13.96	–	–	–	–	535.48	2.75	–	–	–	–	10.91	4.51

1.36% and 2.75%, respectively; for ILS the average gaps are 3.09% and 4.51%, respectively. Both algorithms are once again vastly superior to the 5 greedy heuristics.

Note that all of the gaps from the upper bound are much smaller than for the Group 1 instances. This is because when the reward for all nodes $w_i = 1$, all nodes are equally important, so the algorithm improves a current solution only by finding a new route with more nodes (or the same number of nodes but with lower total distance). In contrast, when the reward w_i for each node is different, then a better solution need not have more nodes than the previous solution. As a result, the incumbent solution can be more easily replaced, which increases the diversity of the neighbourhood. In this sense, the Group 1 instances are more difficult than the Group 2 instances.

It is worth mentioning that when the number of vehicles K is increased by one compared to the best solutions found by Groër (2008), our MA approach is able to find solutions that visit all nodes for all instances (which are optimal in terms of the objective for mVRPP). This shows that the number of vehicles is a crucial factor in determining the difficulty of mVRPP instances.

The detailed results for all experiments described in this section can be obtained from the online supplement to this paper at http://www.computational-logistics.org/orlib.

6. Conclusion

In this paper, we studied a multiperiod vehicle routing problem with profit (mVRPP) that is motivated by the inspector scheduling issues faced by a buying office for one of the largest retailers in the world. In the mVRPP, we have to take into account a regular working time constraint that places a limit on an inspector's daily workload; this problem is a generalization of the team orienteering problem (TOP). We proposed a memetic algorithm (MA) to resolve the problem and conducted a series of computational experiments on various data sets to analyze its effectiveness. The results show that our MA approach produces high-quality solutions that are close to optimal for many cases, and is far superior to what a seasoned scheduler produces.

The algorithm described in this paper forms part of a decision support system that is currently employed by decision-makers of the said buying office. However, as with most practical problems, there are numerous other factors to consider. Examples include:

1. It is not necessarily the case that an inspector must conduct the inspection immediately upon arrival, i.e. it is possible that an inspector first travels to an inspection site, stay overnight in the vicinity of the site, and then perform the inspection at the start of the next period.
2. Different inspectors have different skill sets. For example, one inspector may only be qualified to inspect textiles, while another can inspect both textiles and electronics.
3. Senior or experienced inspectors typically conduct inspections rather quickly if they are familiar with the products. A less experienced inspector can increase their competency by conducting a set of similar inspection tasks. We can apply a learning/forgetting modeling (Hancock and Bayha, 1992) to generate inspection schedules that can improve inspector familiarity with sets of products.
4. As product inspection is one of the critical processes of import/export activities, inspection schedules affect shipment schedules. If a shipment is delayed as a result, buyers might have to incur a loss on profit due to insufficient supply. Therefore, it is worthwhile to consider including a time-dependent reward for the timely completion of inspections.
5. When inspectors travel to and conduct inspections at various sites, schedules may sometimes change unexpectedly due to uncertainties in travel and inspection times. A quick recourse action could be devised to adjust the remaining schedule, which can minimize the inconvenience for both inspectors and suppliers.

In practice, the solution we provided to the mVRPP can be adjusted to handle the above-mentioned examples by having one or more inspectors perform overtime duties. But while this is an adequate practical solution, it would be worthwhile to investigate techniques that explicitly consider these factors.

Acknowledgment

This research was partially supported by NSFC Grant No. 71201065.

References

Archetti, C., Hertz, A., Speranza, M., 2007. Metaheuristics for the team orienteering problem. Journal of Heuristics 13, 49–76.

Beasley, J., 1983. Route-first cluster-second methods for vehicle routing. Omega 11, 403–408.

Bellman, R., 1962. Dynamic programming treatment of the travelling salesman problem. Journal of ACM 9, 61–63.

Bostel, N., Dejax, P., Guez, P., Tricoire, F., 2008. Multiperiod planning and routing on a rolling horizon for field force optimization logistics. In: Golden, B., Raghavan, S., Wasil, E. (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges, vol. 43. Springer, US, pp. 503–525.

Bouly, H., Dang, D.-C., Moukrim, A., 2010. A memetic algorithm for the team orienteering problem. 4OR: A Quarterly Journal of Operations Research 8, 49–70.

Campbell, A.M., Savelsbergh, M., 2004. Efficient insertion heuristics for vehicle routing and scheduling problems. Transportation Science 38, 369–378.

Chao, I.-M., 1993. Algorithms and Solutions to Multi-level Vehicle Routing Problems. Ph.D. Thesis, College Park, MD, USA.

Chao, I.-M., Golden, B.L., Wasil, E.A., 1996. The team orienteering problem. European Journal of Operational Research 88, 464–474.

Christofides, N., Beasley, J.E., 1984. The period routing problem. Networks 14, 237–256.

Chu, F., Labadi, N., Prins, C., 2006. A scatter search for the periodic capacitated arc routing problem. European Journal of Operational Research 169, 586–605.

Cordeau, J.-F., Gendreau, M., Laporte, G., 1997. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 30, 105–119.

Goel, A., 2009. Vehicle scheduling and routing with drivers' working hours. Transportation Science 43, 17–26.

Goel, A., 2010. Truck driver scheduling in the European Union. Transportation Science 44, 429–441.

Golden, B., Raghavan, S., Wasil, E. (Eds.), 2008. The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, US.

Golden, B., Wasil, E., Kelly, J., Chao, I.-M., 1998. The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: Crainic, T., Laporte, G. (Eds.), Fleet Management and Logistics. Kluwer, Boston, pp. 33–56.

Groër, C., 2008. Parallel and Serial Algorithms for Vehicle Routing Problems. Ph.D. Thesis, University of Maryland, College Park, MD.

Hancock, W., Bayha, F., 1992. The learning curve. In: Salvendy, G. (Ed.), Handbook of Industrial Engineering, second ed. John Wiley, New York.

Ke, L., Archetti, C., Feng, Z., 2008. Ants can solve the team orienteering problem. Computers and Industrial Engineering 54, 648–665.

Kok, A.L., Meyer, C.M., Kopfer, H., Schutten, J.M.J., 2010. A dynamic programming heuristic for the vehicle routing problem with time windows and european community social legislation. Transportation Science 44, 442–454.

Laporte, G., 2007. What you should know about the vehicle routing problem. Naval Research Logistics (NRL) 54, 811–819.

Mendoza, J., Medaglia, A., Velasco, N., 2009. An evolutionary-based decision support system for vehicle routing: the case of a public utility. Decision Support Systems 46, 730–742.

Moscato, P., 1999. Memetic algorithms: a short introduction. In: Corne, D., Dorigo, M., Glover, F. (Eds.), New Ideas in Optimization. McGraw Hill, pp. 219–234.

Nagata, Y., Bräysy, O., 2009. Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. Networks 54, 205–215.

Oliver, I., Smith, D., Holland, J., 1987. A study of permutation crossover operators on the traveling salesman problem. In: JJ, G. (Ed.), Proceedings of the Second International Conference on Genetic Algorithms. Lawrence Erlbaum, Hillsdale, NJ, pp. 224–230.

Powell, W.B., Snow, W., Cheung, R.K., 2000. Adaptive labeling algorithms for the dynamic assignment problem. Transportation Science 34, 50–66.

Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers and Operations Research 31, 1985–2002.

Prins, C., 2009. Two memetic algorithms for heterogeneous fleet vehicle routing problems. Engineering Applications of Artificial Intelligence 22, 916–928.

Qin, H., Lim, A., Xu, D., 2009. The selective traveling salesman problem with regular working time windows. In: Chien, B.-C., Hong, T.-P. (Eds.), Opportunities and Challenges for Next-Generation Applied Intelligence, Studies in Computational Intelligence, vol. 214. Springer, Berlin/Heidelberg, pp. 291–296.
Savelsbergh, M., Sol, M., 1998. Drive: dynamic routing of independent vehicles. Operations Research 46, 474–490.
Souffriau, W., Vansteenwegen, P., Berghe, G.V., Oudheusden, D.V., 2010. A path relinking approach for the team orienteering problem. Computers and Operations Research 37, 1853–1859.
Tan, K., Cheong, C., Goh, C., 2007. Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation. European Journal of Operational Research 177, 813–839.
Tang, H., Miller-Hooks, E., 2005. A tabu search heuristic for the team orienteering problem. Computers and Operations Research 32, 1379–1407.
Toth, P., Vigo, D., 2002. The vehicle routing problem. SIAM. Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia.
Vansteenwegen, P., Souffriau, W., Oudheusden, D.V., 2011. The orienteering problem: a survey. European Journal of Operational Research 209, 1–10.
Velasco, N., Castagliola, P., Dejax, P., Guéret, C., Prins, C., 2009. A memetic algorithm for a pick-up and delivery problem by helicopter. In: Pereira, F., Tavares, J. (Eds.), Bio-inspired Algorithms for the Vehicle Routing Problem, Studies in Computational Intelligence, vol. 161. Springer, Berlin/Heidelberg, pp. 173–190.
Villegas, J., Prins, C., Prodhon, C., Medaglia, A., Velasco, N., 2010. GRASP/VND and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots. Engineering Applications of Artificial Intelligence 23, 780–794.
Xu, H., Chen, Z.-L., Rajagopal, S., Arunapuram, S., 2003. Solving a practical pickup and delivery problem. Transportation Science 37, 347–364.