Computers & Industrial Engineering 114 (2017) 195–205

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Solving the team orienteering problem with time windows and mandatory visits by multi-start simulated annealing

Shih-Wei Lin[a,b,c], Vincent F. Yu[d,*]

[a] Department of Information Management, Chang Gung University, Taoyuan, Taiwan
[b] Department of Industrial Engineering and Management, Ming Chi University of Technology, Taipei, Taiwan
[c] Department of Neurology, Chang Gung Memorial Hospital, Linkou, Taoyuan, Taiwan
[d] Department of Industrial Management, National Taiwan University of Science and Technology, Taipei, Taiwan

ARTICLE INFO

Keywords:
Simulated annealing
Team orienteering problem
Mandatory visit
Time window
Multi-start

ABSTRACT

This study investigates the team orienteering problem with time windows and mandatory visits (TOPTW-MV), a new variant of the well-known team orienteering problem with time windows. In TOPTW-MV, some customers are important customers that must be visited. The other customers are called optional customers. Each customer carries a positive score. The goal is to determine a given number of paths to maximize the total score collected at visited nodes, while observing side constraints such as mandatory visits and time window constraints.

We constructed a mathematical programming model and designed a multi-start simulated annealing (MSA) heuristic for TOPTW-MV. Computational study showed that MSA outperforms Gurobi on solving small-scale benchmark instances. Among the 72 small TOPTW-MV instances, MSA obtained better solutions than Gurobi for 13 instances and the same solutions as those obtained by Gurobi for the remaining instances. Moreover, the average computational time of MSA is shorter than that of Gurobi. In addition, computational study based on 168 TOPTW-MV benchmark instances adapted from existing TOPTW benchmark instances indicated that MSA significantly improves the performance of basic simulated annealing heuristic and outperforms the artificial bee colony algorithm on solving large TOPTW-MV instances.

1. Introduction

The orienteering problem (OP) was inspired by a sports game in which several players start from a fixed control point, visit some check points, each carries a score, and then return to the control point within a given time limit (Campos, Marti, Sanchez-Oro, & Duarte, 2014; Chekuri, Korula, & Pal, 2012; Fischetti, Gonzalez, & Toth, 1998; Golden, Levy, & Vohra, 1987; Gunawan, Lau, & Vansteenwegen, 2016; Leifer & Rosenwein, 1994; Muthuswamy & Lam, 2011; Vansteenwegen, Souffriau, & Van Oudheusden, 2011; Wang, Sun, Golden, & Jia, 1995). The player who collects the most scores wins the game. In network terms, there are a starting node, an end node, and a set of nodes that can be visited, each with a positive score and a positive service time. Both of the starting node and the end node correspond to the control point. The goal is to find a path from the starting node to the end node that maximizes the total score collected from the visited nodes. The length of the path, including travel times on the arcs and service times at the visited nodes, cannot exceed a given time limit. The orienteering problem with time windows (OPTW) is a natural extension of OP when visits must start within a given time window associated with each visited node (Duque, Lozano, & Medaglia, 2015; Gunawan, Lau, & Lu, 2015; Kantor & Rosenwein, 1992; Mann, Zion, Rubinstein, Linker, & Shmulevich, 2016; Righini & Salani, 2009).

OP and OPTW have many applications in logistics, tourism, and other fields. Therefore, they have received much attention in recent years (Gunawan et al., 2016; Vansteenwegen et al., 2011). When multiple paths are considered, OP and OPTW become the team orienteering problem (TOP) (Archetti, Hertz, & Speranza, 2007; Chao, Golden, & Wasil, 1996; Dang, Guibadj, & Moukrim, 2013; El-Hajj, Dang, & Moukrim, 2016; Ke, Zhai, Li, & Chan, 2016; Keshtkaran, Ziarati, Bettinelli, & Vigo, 2016; Kim, Li, & Johnson, 2013; Souffriau, Vansteenwegen, Vanden Berghe, & Van Oudheusden, 2010; Tang & Miller-Hooks, 2005; Vansteenwegen, Souffriau, Vanden Berghe, & Van Oudheusden, 2009a) and the team orienteering problem with time windows (TOPTW) (Gedik, Kirac, Milburn, & Rainwater, 2017; Hu & Lim, 2014; Labadie, Mansini, Melechovský, & Wolfler Calvo, 2012; Labadie, Melechovský, & Wolfler Calvo, 2010; Lin & Yu, 2012, 2015; Souffriau, Vansteenwegen, Vanden Berghe, & Van Oudheusden, 2013; Tae & Kim, 2015; Vansteenwegen, Souffriau, Vanden Berghe, & Van Oudheusden, 2009b). This study investigates the

* Corresponding author.
E-mail address: vincent@mail.ntust.edu.tw (V.F. Yu).

http://dx.doi.org/10.1016/j.cie.2017.10.020
Received 2 May 2017; Received in revised form 18 October 2017; Accepted 19 October 2017
Available online 20 October 2017
0360-8352/

team orienteering problem with mandatory visits (TOPTW-MV), a new extension of TOPTW in which some of the nodes, called mandatory nodes, must be visited. The other nodes are called optional nodes. Each of the mandatory nodes and optional nodes carries a positive score. The goal is to maximize the scores collected from visited nodes. Applications of TOPTW-MV include planning road maintenance activities, designing tourist trips, routing waste collecting vehicles. To the best of our knowledge, this problem has not been studied before. We formulated a mixed integer linear programming (MILP) model for the problem and developed a multi-start simulated annealing (MSA) heuristic for solving the problem.

The rest of the paper is organized as follows. Section 2 reviews relevant studies. Section 3 presents a mixed integer linear programming model for TOPTW-MV. Section 4 describes the proposed multi-start simulated annealing heuristic. Section 5 discusses experimental results. Section 6 gives conclusions and suggestions for future research.

2. Literature review

Several TOPTW-MV related problems have been investigated. Gendreau, Laporte, and Semet (1998) studied the undirected selected traveling salesman problem (STSP), which can be regarded as OP with mandatory visits. The problem is defined on a network where each node is associated with a profit, while each edge is associated with a cost. Some of the nodes are compulsory and must be visited. The goal is to find a tour that maximizes total profit while all the compulsory nodes are visited and the total cost does not exceed a given constant. They developed a branch-and-cut algorithm for the problem and used it to solve instances with up to 300 nodes. Tricoire, Romauch, Doerner, and Hartl (2010) studied the multi-period orienteering problem with time windows, in which it is mandatory to visit long-term customers during regular periods; while optional to visit potential customers. Every customer has several time windows which may be different from day to day. The authors hybridized a variable neighborhood search algorithm with an exact procedure to solve the problem.

Salazar-Aguilar, Langevin, and Laporte (2014) introduced the multi-district team orienteering problem (MDTOP) which comes from a real case of periodic road maintenance activities planning faced by the Ministry of Transport of Québec province, Canada. The maintenance region is divided into several districts, each associated with certain mandatory and optional activities. The authors formulated a MILP model for the problem and proposed an adaptive large neighborhood search algorithm for solving MDTOP. Inspired by the work of Salazar-Aguilar et al. (2014), Palomo-Martínez, Salazar-Aguilar, Laporte, and Langevin (2017) proposed the orienteering problem with mandatory visits and exclusionary constraints, which considers mandatory visits and incompatibilies among nodes. They hybridized a reactive GRASP and a general variable neighborhood search (VNS) to solve the problem.

Since OP is NP-hard (Golden et al., 1987), many solution approaches have been proposed for solving OP and its extensions based on popular optimization techniques such as constraint programming (Gedik et al., 2017) and branch-and-price approaches (Tae & Kim, 2015), and metaheuristics including GRASP (Campos et al., 2014; Souffriau et al., 2013), iterated local search (Gunawan et al., 2015; Souffriau et al., 2013; Vansteenwegen et al., 2009b), particle swarm optimization (Dang et al., 2013; Muthuswamy & Lam, 2011; Yu, Jewpanya, Ting, & Redi, 2017), path-relinking (Campos et al., 2014; Souffriau et al., 2010), simulated annealing (SA) (Lin & Yu, 2012, 2015), tabu search (Tang & Miller-Hooks, 2005), variable neighborhood search (Kim et al., 2013; Palomo-Martínez et al., 2017), and other sophisticated metaheuristics (Hu & Lim, 2014; Ke et al., 2016; Labadie et al., 2010; Vansteenwegen et al., 2009a; Wang et al., 1995). This study proposed a multi-start simulated annealing heuristic for solving TOPTW-MV because it performs very well on solving TOP (Lin, 2013).

3. MILP model

The TOPTW with mandatory visits can be defined on a complete directed network $G = (N,A)$, where $N = \{0,1,2,...,n\}$ is the set of nodes; $A = \{(i,j): i \neq j, i,j \in N\}$ is the set of arcs, each of which connects a pair of nodes. Node 0 is designated as the depot. Each of the other nodes i ($i = 1,\ldots, n$) represents a customer (location) and has a nonnegative score S_i, a nonnegative service time T_i, and a time window $[O_i, C_i]$ during which the service to the node can start. Each path must start from and end at node 0. The time needed to travel from node i to node j is known as t_{ij} for each pair of nodes i and j. The length of a path (measured in time) cannot exceed the pre-defined time limit T_{max}. Every node can be visited at most once. Some of the customers are important customers and thus must be serviced. Waiting at a customer location until the opening of the customer's time window to start service is allowed. The maximum number of paths should not exceed the number of important customers. Each path, if used, incurs a given constant cost. The objective is maximizing the total net profit, which is calculated by subtracting the total cost of paths from the total score collected from visiting nodes. TOPTW-MV can be modeled as a mixed integer linear program as follows.

Decision Variables:

$y_{ik} = 1$ if customer i is visited in path k, 0 otherwise
$x_{ijk} = 1$ if, in path k, a visit to customer i is followed by a visit to customer j, 0 otherwise
s_{ik} = starting time of the service to customer i in path k
$M_k = 1$ if path k is used, 0 otherwise

Parameters:

n: number of customers
γ: constant cost of a path
P: maximum number of paths
v_i: if customer i is an important customer, $v_i = 1$; otherwise $v_i = 0$
s_i: score of customer i
T_{max}: time budget of a path
t_{ij}: time required to travel from customer i to customer j
T_i: service time required for customer i
O_i: opening time of the time window for customer i
C_i: closing time of the time window for customer i
L: a large constant

Model:

$$Maximize \sum_{k=1}^{P}\sum_{i=1}^{n} s_i y_{ik} - \gamma \sum_{k=1}^{P} M_k \quad (1)$$

s.t.

$$\sum_{j=1}^{n} x_{0jk} = \sum_{j=1}^{n} x_{j0k} = M_k \quad \forall\, k = 1,2,...,P \quad (2)$$

$$\sum_{k=1}^{P} y_{ik} \leqslant 1 \quad \forall\, i = 1,2,...,n \quad (3)$$

$$\sum_{k=1}^{P} y_{ik} \geqslant v_i \quad \forall\, i = 1,2,...,n \quad (4)$$

$$\sum_{\substack{i=0 \\ i \neq l}}^{n} x_{ilk} = \sum_{\substack{j=0 \\ j \neq l}}^{n} x_{ljk} = y_{lk} \quad \forall\, l = 1,...,n\ ;\ \forall\, k = 1,...,P \quad (5)$$

$$s_{ik} + T_i + t_{ij} - s_{jk} \leqslant L(1 - x_{ijk}) \quad \forall\, i,j = 1,...,n\ ;\ \forall\, k = 1,2,...,P \quad (6)$$

$$\sum_{i=0}^{n}\left(T_i y_{ik} + \sum_{j=0}^{n} t_{ij} x_{ijk}\right) \leqslant T_{\max} \quad \forall\, k = 1,2,\ldots,P \tag{7}$$

$$O_i y_{ik} \leqslant s_{ik} \quad \forall\, i = 1,\ldots,n\,; \quad \forall\, k = 1,\ldots,P \tag{8}$$

$$s_{ik} \leqslant C_i y_{ik} \quad \forall\, i = 1,\ldots,n\,; \forall\, k = 1,\ldots,P \tag{9}$$

$$x_{ijk}, y_{ik} \in \{0,1\} \quad \forall\, i,j = 0,\ldots,n\,; \forall\, k = 1,\ldots,P \tag{10}$$

The objective function (1) maximizes the total profit. Constraint (2) determines the number of paths used. Constraints (3) make sure that every node is visited at most once. Constraints (4) ensure mandatory visits. Constraints (5) and (6) maintain the connectivity and keep track of the timeline of each path. Constraints (7) guarantee that time budget of each path is not violated. Constraints (8) and (9) are time window constraints. Constraints (10) specify that all variables are binary.

4. Multi-start simulated annealing algorithm

This section describes the representation of the solution and the calculation of the value of the objective function for the proposed MSA algorithm and then discusses others parts of the MSA algorithm that is used herein to solve the TOPTW-MV of interest. The MSA developed for TOPTW-MV in this study is similar to the one for TOP (Lin, 2013). However, MSA for TOPTW-MV requires fewer parameters. More specifically, the threshold parameter is not used to speed up the algorithm since preliminary experiments indicate that its impact on the performance of MSA is insignificant. The MSA used for solving TOPTW-MV is modified to reflect the change.

4.1. Solution representation

A solution of TOPTW-MV is specified by a string of numbers that consists of a permutation of n customers (1,2,...,n) and $P-1$ 0's (paths), where P is the maximum number of paths. The zeros are employed to separate paths. Therefore, the total number of elements in a solution is $n+P-1$. The i^{th} non-zero number in the $n+P-1$ positions denotes the i^{th} customer to be serviced. Therefore, the first non-zero entry in the solution representation indicates the first customer in the first path. Then, from left to right, other customers are appended to the tour one at a time, under the condition that time window constraint at each node is not violated.

If adding a customer to the current path violates the time window constraint of the customer or the depot, then the path skips this customer and considers the next customer. If a vehicle arrives before the customer's time window, then the service is delayed until the

Table 1
A TOPTW-MV example with 20 customers and three mandatory visits.

No	X	Y	T	S	OT	CT	MV
0	40	50	0	0	1	1236	
1	45	68	90	10	912	967	0
2	45	70	90	30	825	870	0
3	42	66	90	10	65	146	0
4	42	68	90	10	727	782	0
5	42	65	90	10	15	67	0
6	40	69	90	20	621	702	1
7	40	66	90	20	170	225	0
8	38	68	90	20	255	324	1
9	38	70	90	10	534	605	0
10	35	66	90	10	357	410	0
11	35	69	90	10	448	505	0
12	25	85	90	20	652	721	1
13	22	75	90	30	30	92	0
14	22	85	90	10	567	620	0
15	20	80	90	40	384	429	0
16	20	85	90	40	475	528	0
17	18	75	90	20	99	148	0
18	15	75	90	20	179	254	0
19	15	80	90	10	278	345	0
20	30	50	90	10	10	73	0

17	13	18	5	15	14	12	19	2	0	7	3	8	9	6	11	16	4	1	0	20	10

Fig. 1. An example of solution representation.

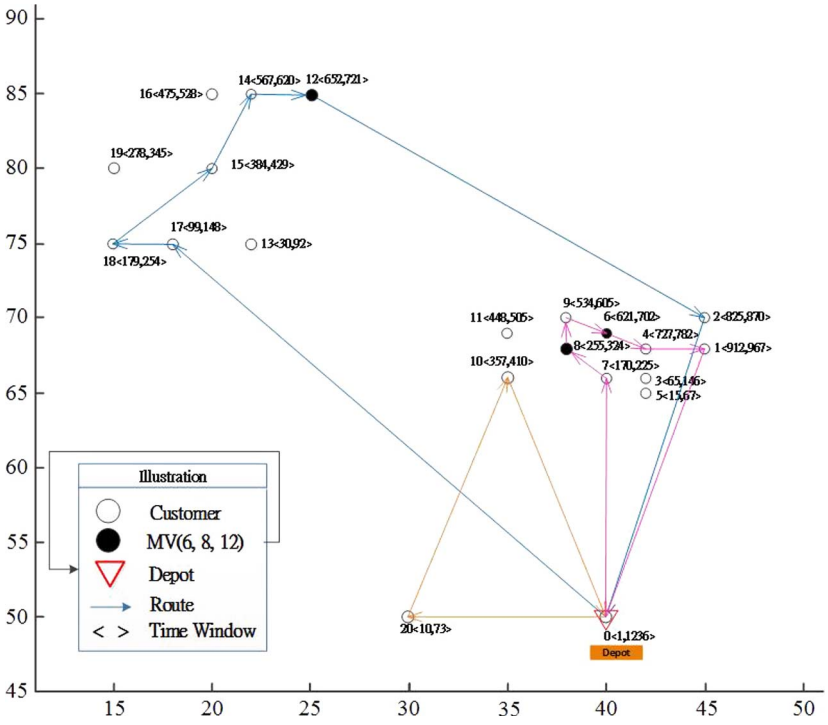

Fig. 2. A visual illustration of the example solution given in Fig. 1.

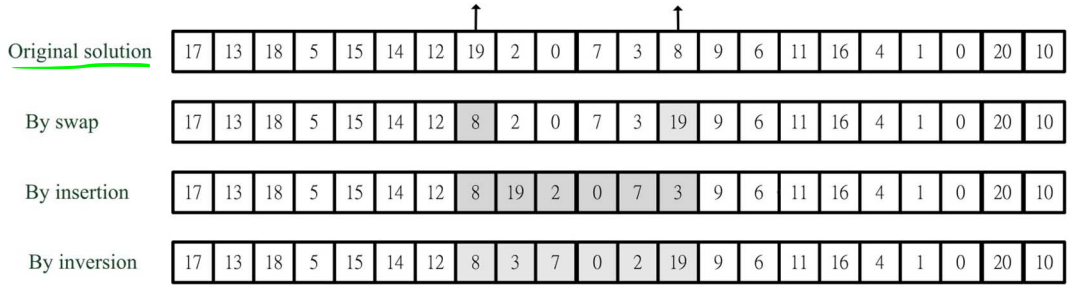

Fig. 3. Examples of neighborhood solutions.

customer's time window starts. A zero in the solution representation terminates current path and starts a new path whenever feasible. Notably, even if several customers have the same service time, the starting and finishing times of their service time window may not be the same.

4.2. Illustration of solution representation

Table 1 gives a TOPTW-MV instance with 20 customers, which can be serviced by two paths. Each customer's coordinates (X, Y), service time (T), score (S), opening time (OT) and closing time (CT) of time window, and indicator of mandatory visit (MV) are listed in the table. If the value in MV column equals to one, then the corresponding customer must be visited; otherwise, a visit to the customer is optional. Because the number of mandatory visits is three, the maximum number of paths is three.

An example of solution to this TOPTW-MV instance is given in Fig. 1, where two ($P - 1 = 3 - 1 = 2$) dummy zeros are introduced. The travel time between each pair of nodes is assumed to be the Euclidean distance between the nodes, rounded down to the first decimal. Fig. 2 visually illustrates the visual illustration of this solution.

As can be seen in Fig. 2, the first customer to be serviced in the first path is customer 17. The path then goes on to service customers 18, 15, 14, 12, and 2. Because the next number in the solution is zero, the path returns to the depot after finishing servicing customer 2. Note that the first path skips customers 13, 5, and 19 because servicing any one of them violates the time window constraint. The first customer serviced by the second path is customer 7. Subsequently, customers 8, 9, 6, 4 and 1 are serviced in this path. After finishing customer 1 is serviced, the path returns to the depot because the next number in the solution is

MSA (I_{iter}, T_0, $N_{non-improving}$, P_{size} and α)
Step 1: Generating the initial solutions σ_k ($k = 1, 2, ..., P_{size}$);
Step 2: Let $T = T_0$; N=0; σ_{best} =the best σ_k among the P_{size} solutions;
$F_{best} = obj(\sigma_{best})$; R=0; FoundBestSol=false;
Step 3: N=N+1;
Step 4: For $k = 1$ to P_{size} {
Step 4.1 Generating a solution σ'_k based on σ_k ;
Generate s~U(0,1);
If ($s \le 1/3$)
Generate σ'_k by Swap;
Else_If ($s > 1/3$ && $s \le 2/3$)
Generate σ'_k by Swap;
Else
Generate σ'_k by Reversion;
Step 4.2 If $\Delta_k = obj(\sigma'_k) - obj(\sigma_k) \ge 0$ {Let $\sigma_k = \sigma'_k$;}
Else {
Generate r~U(0,1);
If $r < e^{\Delta_k / T}$ { Let $\sigma_k = \sigma'_k$;}
}
Step 4.3 If $obj(\sigma_k) > obj(\sigma_{best})$ {
$\sigma_{best} = \sigma_k$; R=0; FoundBestSol=true;
}
}
Step 5: If $N = I_{iter}$ {
$T = T \times \alpha$; $N = 0$;
If (FoundBestSol is false) { R=R+1; FoundBestSol=true;}
}
Else {Go to Step 3;}
Step 6: If $R = N_{non-improving}$ {Terminate the MSA procedure;}
Else {Go to Step 3;}

Fig. 4. The pseudocode of the proposed MSA.

Table 2
Parameter values tested in the calibration.

Parameter	Values			
T_0	1.00	**3.00**	5.00	7.00
α	0.875	**0.900**	0.925	0.950
I_{iter}	2000	3000	4000	**5000**
$N_{non\text{-}improving}$	5	10	**15**	20

Bold numbers indicate the best parameter values.

zero. Customers 3, 11, and 16 are excluded from the path because of violation of time window constraint. The third path services customer 20 and then customer 10.

The total net profit of a given solution X, denoted by TNP(X), can be easily calculated (total score collected in all paths minus total cost of paths). If mandatory visits are not performed, a large penalty is given to the solution, which makes accepting the solution unlikely in the process of searching for the optimal solution.

4.3. Neighborhood solutions

As shown in Fig. 3, the proposed MSA algorithm uses three types of moves (swap, insertion, and inversion) to generate neighborhood solutions. Let X be the current solution and be the set of its neighborhood solutions. A new solution Y is selected from at each iteration by using one of the three moves in the following manner. The swap move exchanges the positions of two randomly selected entries in X. The insertion move randomly selects an entry in X and then inserts it before another randomly selected entry in X. The inversion move randomly selects two entries in X and then reverses the sequence between them (including these two entries).

4.4. MSA procedure

At the beginning, the current temperature T is set to T_0. Then the initial solutions σ_k ($k = 1,...,P_{size}$) are generated at random and used as multi-start points. Let σ_{best} be the best solution found so far as the algorithm proceeds. σ_{best} is set to be the best σ_k ($k = 1,...,P_{size}$) at the beginning. At each iteration, new solutions σ'_k ($k = 1,...,P_{size}$) are selected from their corresponding neighborhood $N(\sigma_k)$. Let $TNP(\sigma_k)$ denote the objective function value of σ_k and Δ_k be the difference between the total net profit of σ_k and σ'_k, that is $\Delta_k = TNP(\sigma'_k) - TNP(\sigma_k)$. The probability of replacing σ_k with σ'_k, given that $\Delta_k < 0$, is $e^{(\Delta_k/T)}$. This is accomplished by generating a random number $r \in [0, 1]$ and replacing σ_k with σ'_k if $r < e^{(\Delta_k/T)}$. If $\Delta_k \geqslant 0$, then the probability of replacing σ_k with σ'_k is 1. $TNP(\sigma_k)$ is then compared with $TNP(\sigma_{best})$. Whenever there is a σ_k such that $TNP(\sigma_{best}) < TNP(\sigma_k)$, then the algorithm updates σ_{best} by setting $\sigma_{best} = \sigma_k$. The current temperature T is reduced to αT ($0 < \alpha < 1$) after I_{iter} iterations. When σ_{best} has not been improved in $N_{non\text{-}improving}$ successive temperature reductions, the algorithm stops. After the MSA procedure terminates, σ_{best} gives an (near) optimal solution. The pseudocode of the proposed MSA is shown in Fig. 4.

5. Experimental results and discussion

The proposed MSA algorithm was implemented in C. The experiments are carry out on a computer with an Intel Core 2 2.67 GHz CPU. The MILP model for TOPTW-MV was solved using Gurobi 6.5 on the same machine.

5.1. Test problems

To demonstrate the applicability of the MSA algorithm, 258 (18 + 72 + 168) problem instances are generated based on Righini and

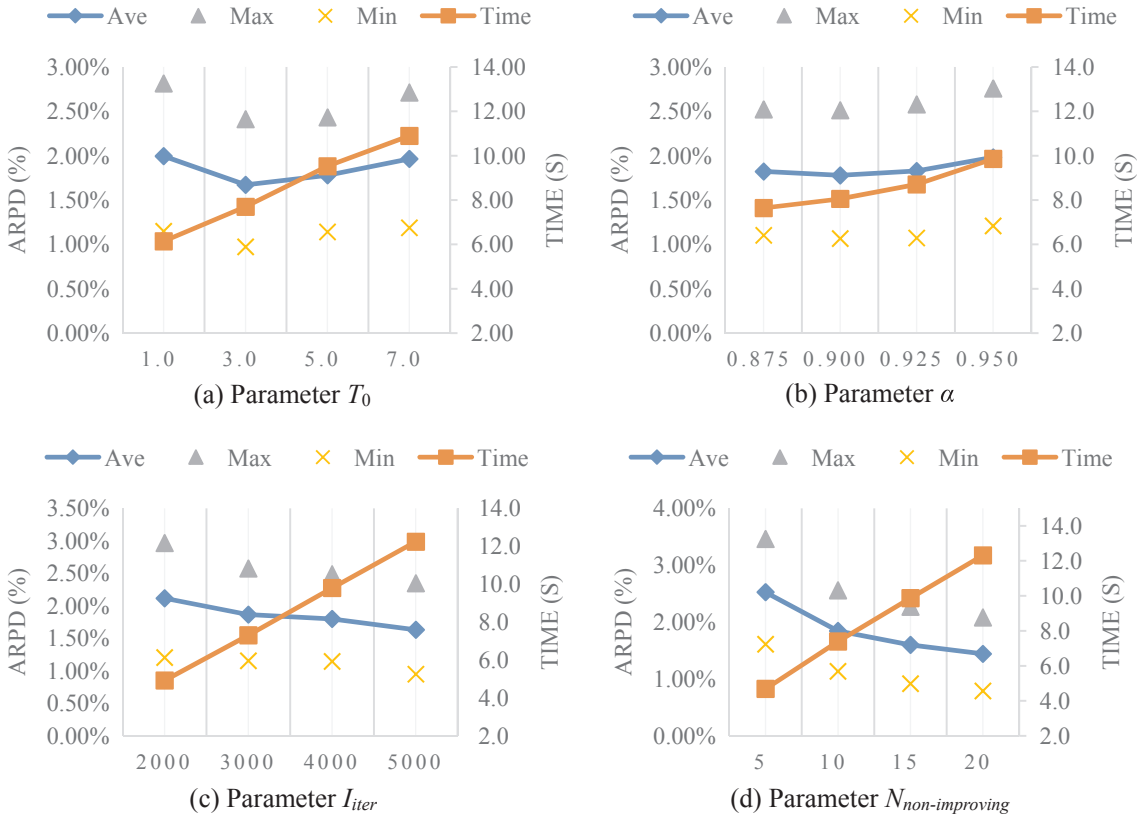

Fig. 5. The effect of each parameter on solution quality.

Table 3
Computational results for small problems.

		Gurobi				MSA (Psize = 3)					Gurobi				MSA (Psize = 3)		
#M	Inst.	LB	TNP	#Path used	Time(s)	TNP	#Path used	Time(s)	#M	Inst.	LB	TNP	#Path used	Time(s)	TNP	#Path used	Time(s)
2	c101	200	**200**	2	0.13	**200**	2	1.73	4	c101	300	**300**	4	0.86	**300**	4	2.11
2	c103	460	340	2	1800.04	350	2	1.91	4	c103	469	340	3	1800.06	340	3	2.03
2	c105	180	**180**	2	4.47	**180**	2	1.74	4	c105	200	**200**	3	57.47	**200**	3	2.25
2	r101	33	**33**	2	0.06	**33**	2	1.49	4	r101	35	**35**	3	0.16	**35**	3	1.78
2	r103	243	190	2	1800.05	190	2	1.84	4	r103	313	159	3	1800.21	169	3	2.27
2	r105	153	**153**	2	0.88	**153**	2	1.69	4	r105	159	**159**	3	19.19	**159**	3	2.07
2	rc101	37	**37**	2	0.20	**37**	2	1.44	4	rc101	155	**155**	4	3.30	**155**	4	1.89
2	rc103	284	240	2	1800.03	240	2	1.73	4	rc103	383	267	3	1800.05	275	3	2.65
2	rc105	155	**155**	2	89.61	**155**	2	1.90	4	rc105	221	160	3	1800.00	160	3	1.98
3	c201	340	**340**	2	0.53	**340**	2	1.80	4	c201	340	**340**	2	1.39	**340**	2	2.06
3	c203	399	370	1	1800.02	370	1	1.77	4	c203	419	320	2	1800.02	370	1	2.06
3	c205	380	**380**	1	6.52	**380**	1	1.89	4	c205	380	**380**	1	32.47	**380**	1	2.14
3	r201	304	**304**	1	4.23	**304**	1	1.90	4	r201	304	**304**	1	9.50	**304**	1	2.11
3	r203	386	287	2	1800.00	337	1	1.74	4	r203	386	337	1	1800.02	337	1	1.93
3	r205	331	**331**	1	70.30	**331**	1	2.17	4	r205	331	**331**	1	452.77	**331**	1	2.27
3	rc201	303	**303**	2	95.91	**303**	2	1.80	4	rc201	303	**303**	2	699.89	**303**	2	2.03
3	rc203	456	357	2	1800.02	357	2	1.78	4	rc203	456	357	2	1800.00	357	2	2.02
3	rc205	362	320	2	1800.13	333	1	1.82	4	rc205	405	320	2	1800.03	320	2	2.04
3	c101	270	**270**	3	1.09	**270**	3	1.92	5	c101	300	**300**	4	0.83	**300**	4	2.41
3	c103	469	340	3	1800.00	350	2	2.72	5	c103	469	340	3	1800.05	340	3	2.17
3	c105	200	**200**	3	18.31	**200**	3	2.21	5	c105	200	**200**	3	11.80	**200**	3	2.35
3	r101	37	**37**	3	0.11	**37**	3	1.92	5	r101	35	**35**	3	0.20	**35**	3	1.89
3	r103	312	169	3	1800.03	169	3	1.89	5	r103	313	149	3	1800.38	169	3	2.81
3	r105	165	**165**	3	6.02	**165**	3	2.38	5	r105	157	**157**	4	27.72	**157**	4	4.67
3	rc101	125	**125**	3	0.89	**125**	3	1.69	5	rc101	152	**152**	4	7.69	**152**	4	2.04
3	rc103	383	275	3	1800.00	279	3	3.52	5	rc103	383	254	4	1800.02	275	3	3.08
3	rc105	181	160	3	1800.02	160	3	1.90	5	rc105	232	150	4	1800.02	155	3	2.68
2	c201	340	**340**	2	0.83	**340**	2	1.91	5	c201	340	**340**	2	1.95	**340**	2	2.14
2	c203	399	370	1	1800.02	370	1	1.92	5	c203	419	370	1	1800.02	370	1	2.14
2	c205	380	**380**	1	8.97	**380**	1	1.94	5	c205	380	**380**	1	54.61	**380**	1	2.40
2	r201	304	**304**	1	18.56	**304**	1	3.15	5	r201	304	**304**	1	23.66	**304**	1	2.45
2	r203	386	337	1	1800.02	337	1	1.83	5	r203	386	287	2	1800.02	337	1	2.12
2	r205	331	**331**	1	19.41	**331**	1	1.87	5	r205	331	**331**	1	1472.08	**331**	1	2.22
2	rc201	303	**303**	2	350.97	**303**	2	1.90	5	rc201	303	**303**	2	1061.16	**303**	2	2.15
2	rc203	456	357	2	1800.00	357	2	1.91	5	rc203	456	357	2	1800.02	357	2	2.14
2	rc205	405	342	1	1800.03	342	1	3.76	5	rc205	405	320	2	1800.08	342	1	3.57

Bold numbers indicate optimal solution values.

Salani (2009). These problems can be classified as the problems for parameter calibration, the small problems, and the large problems. First, 18 problem instances are used to calibrate the parameters. To compare MILP with MSA, 72 small problem instances are used. After that, 168 large problem instances are used to compare the performance of MSA, SA, and the artificial bee colony (ABC) algorithms. The ABC used in the comparison is based on the algorithm proposed by Cura (2014).

Parameter calibration uses 18 problem instances in which the number of customers ranges from 25 to 100, and the number of mandatory visits varies from two to five. For the 72 small problem instances, the number of customers is 25, and the number of mandatory visits ranges from two to five. For large problems, the number of customers is 100 and the number of mandatory visits is set to be five.

5.2. Parameter calibration

Since the computational results of the MSA algorithm are affected by the parameter settings, parameter calibration used 18 problem instances. In the experiment, several values of the parameters T_0, I_{iter}, α, and $N_{non\text{-}improving}$ (Table 2) are used. Each parameter has four levels so 256 combinations of parameter values were generated.

For each combination of parameter values, each of the 18 problem instances was solved three times. The maximum, average, and minimum total net profit for each problem are recorded and the *RPD* (relative percentage deviation) is calculated as $RPD_{BKS} = (TNP^{BKS}_{\max} - TNP^{P}_{\max})/TNP^{BKS}_{\max} \times 100\%$, where $TNP^{P}_{\max}$ is the total net profit with the parameter combination *P and* $TNP^{BKS}_{\max}$ is the total net profit of the best solution that was obtained by the proposed MSA heuristic in the parameter analysis. Fig. 3 presents the effect of each parameter on solution quality.

As presented in Fig. 5(a), if the initial temperature T_0 is too high, then some excessive computational time is required to find better solutions. If T_0 is too low, then the proposed algorithm may converge prematurely because the probability for accepting inferior solutions is low. A higher α requires more time to reduce the temperature to the stopping value if the proposed algorithm stops at the final temperature. Because one of the stopping conditions for the algorithm is $N_{non\text{-}improving}$, the range of computational time is not large for various α values, as presented in Fig. 5(b). As I_{iter} increases, better solutions were obtained using more computational time, as shown in Fig. 5(c). As shown in Fig. 5(d), increasing $N_{non\text{-}improving}$ improves the quality of the solution at the cost of more computational time. Considering the solution quality and computational time required, we set $T_0 = 3.0$, $\alpha = 0.90$, $I_{iter} = 5000$, and $N_{non\text{-}improving} = 15$ in the proposed algorithm. P_{size} is set to be 1, 2, 3, 4, 5, and 6 for testing the MSA. After pre-testing, $P_{size} = 3$ obtained the best result, and is thus used in this study.

The parameters for the artificial been colony algorithm are based on the parameters used for solving TOPTW (Cura, 2014). However, for comparison purpose, the execution time (termination condition) is adjusted so that it is about the same as that of MSA. The parameter tuning process is similar to the one used by Cura (2014).

Table 4
Computational results for problems with 5 mandatory visits and a constant path cost of 75.

Instance	MSA				SA				ABC			
	Best	#visited	#path	time(s)	Best	#visited	#path	time(s)	Best	#visited	#path	time(s)
c101	805	50	5	28.34	**815**	5	51	29.44	**815**	51	5	27.54
c102	**945**	56	5	30.76	935	5	55	26.18	**945**	56	5	32.49
c103	**995**	56	5	29.82	985	5	56	28.31	**995**	57	5	29.56
c104	1005	57	5	25.37	1015	5	58	32.50	**1025**	59	5	37.03
c105	**855**	52	5	33.52	845	5	51	28.28	**855**	52	5	31.06
c106	**875**	52	5	30.17	**875**	5	52	31.04	**875**	52	5	38.9
c107	**905**	54	5	30.77	**905**	5	54	25.65	**905**	54	5	34.05
c108	915	55	5	33.95	915	5	55	35.72	**925**	56	5	38.8
c109	**1015**	59	5	38.57	985	5	56	29.01	**1015**	59	5	45.91
r101	**310**	36	5	23.82	308	5	34	31.38	304	34	5	24.52
r102	560	49	5	35.83	557	5	48	25.62	**566**	51	5	37.97
r103	654	52	5	36.66	651	5	53	34.47	**657**	54	5	42.85
r104	**728**	57	5	32.73	709	5	57	31.80	**737**	56	5	52.95
r105	504	43	5	27.99	506	5	43	27.87	**511**	45	5	32.54
r106	**655**	51	5	26.07	643	5	50	32.76	649	50	5	47.56
r107	**670**	54	5	30.28	667	5	52	25.63	669	54	5	51.63
r108	**736**	56	5	39.73	728	5	55	28.34	728	55	5	50.77
r109	616	49	5	27.54	**621**	5	50	25.62	617	50	5	34.82
r110	650	51	5	26.44	**651**	5	52	32.97	650	51	5	47.93
r111	680	56	5	32.73	**684**	5	53	35.35	676	54	5	42.28
r112	745	56	5	35.75	**748**	5	56	29.74	743	55	5	50.68
rc101	**583**	45	5	22.71	**583**	5	45	20.51	**583**	45	5	33.31
rc102	**693**	50	5	26.39	689	5	51	27.48	683	48	5	34.63
rc103	**755**	51	5	29.81	740	5	49	26.38	**755**	50	5	55.21
rc104	846	54	5	28.77	**851**	5	53	34.05	847	53	5	38.1
rc105	644	45	5	31.17	651	5	48	25.63	**656**	48	5	36.39
rc106	**666**	50	5	29.73	**666**	5	50	25.40	661	48	5	41.92
rc107	**730**	49	5	29.81	724	5	49	33.36	724	50	5	36.69
rc108	832	52	5	40.16	828	5	51	27.00	**839**	53	5	50.29
c201	1510	100	4	32.55	**1515**	3	93	34.51	1510	100	4	25.5
c202	**1525**	94	3	40.57	1510	4	100	32.08	1510	100	4	26.9
c203	**1510**	100	4	28.36	**1510**	4	100	27.98	**1510**	100	4	24.83
c204	**1510**	100	4	20.21	**1510**	4	100	23.54	**1510**	100	4	25.75
c205	**1535**	95	3	29.07	**1535**	3	95	30.66	1510	100	4	22.5
c206	1515	93	3	25.84	**1525**	3	94	28.96	1510	100	4	23.99
c207	**1545**	96	3	32.78	1510	4	100	25.61	1435	100	5	21.62
c208	1510	100	4	22.37	**1525**	3	94	30.10	1510	100	4	21.16
r201	1183	93	3	46.28	**1198**	3	93	48.13	1158	100	4	30.62
r202	**1226**	97	3	36.72	**1226**	3	97	37.99	1158	100	4	22.44
r203	**1233**	100	3	35.06	**1233**	3	100	29.23	1158	100	4	23.02
r204	**1158**	100	4	21.36	**1158**	4	100	20.81	1158	100	4	19.96
r205	**1233**	100	3	37.39	**1233**	3	100	35.79	1158	100	4	20.11
r206	1227	97	3	27.66	**1233**	3	100	36.06	1158	100	4	22.27
r207	1230	98	3	23.54	**1233**	3	100	24.64	**1233**	100	3	25.58
r208	**1233**	100	3	20.17	1158	4	100	20.01	1158	100	4	19.38
r209	1218	94	3	26.81	**1233**	3	100	41.53	1158	100	4	23.96
r210	**1233**	100	3	34.06	1231	3	99	29.87	1158	100	4	20.56
r211	**1233**	100	3	24.52	**1233**	3	100	26.90	1158	100	4	20.83
rc201	1434	92	3	47.05	**1448**	3	93	39.48	1424	100	4	24.27
rc202	**1489**	99	3	34.31	1484	3	98	33.67	1424	100	4	21.51
rc203	**1424**	100	4	21.15	**1424**	4	100	26.01	**1424**	100	4	20.45
rc204	**1424**	100	4	24.69	**1424**	4	100	24.39	**1424**	100	4	19.22
rc205	1449	92	3	38.65	**1459**	3	94	38.84	1424	100	4	23.98
rc206	**1494**	98	3	34.49	1424	4	100	28.58	1424	100	4	23.03
rc207	**1424**	100	4	29.22	**1424**	4	100	29.63	**1424**	100	4	19.94
rc208	**1424**	100	4	22.20	**1424**	4	100	25.51	**1424**	100	4	19.14
#BKS	**33**				31				25			
Gap	**0.364**				0.706				1.550			
Time(s)	30.58				30.98				31.97			

Bold numbers indicate the best solution values found by MSA, SA, or ABC.

5.3. Comparison of results obtained by MSA and Gurobi solver

For comparison of MSA results with those obtained from solving MILP by Gurobi solver; each problem is solved by MSA only once because its performance is consistent. When MSA is used to solve small problems, the results from different runs are about the same. More specifically, the average solution value from 5 runs of the algorithm is merely 0.26% off from the best solution value on average. Gurobi 6.5 was used to solve the model. The solver was terminated after 1800 s or when it had found an optimal solution. Table 3 presents solutions to the 72 small problem instances and the corresponding computational times. The Gurobi solver finds optimal solutions to 41 problems, as indicated in bold in Table 3. MSA also generates optimal solutions to these 41 problems. MSA (with P_{size} = 3) yields better solutions than the Gurobi solver to 13 problems among the remaining problems.

Many factors may influence the computational time, including CPU

Table 5
Computational results for problems with 5 mandatory visits and a constant path cost of 100.

Instance	MSA				SA				ABC			
	Best	#visited	#path	time(s)	Best	#visited	#path	time(s)	Best	#visited	#path	time(s)
c101	680	50	5	28.64	670	50	5	30.24	**690**	51	5	32.48
c102	810	56	5	28.12	**820**	56	5	29.19	**820**	56	5	34.96
c103	860	56	5	30.54	860	56	5	30.35	**870**	57	5	40.81
c104	890	58	5	26.48	890	58	5	31.61	**900**	59	5	29.81
c105	710	50	5	27.49	720	50	5	31.42	**730**	52	5	32.16
c106	**750**	52	5	28.90	**750**	53	5	34.03	**750**	52	5	37.36
c107	**780**	54	5	26.48	**780**	54	5	29.89	**780**	54	5	31.20
c108	780	54	5	28.18	**790**	55	5	28.97	**790**	55	5	30.91
c109	880	58	5	36.03	870	57	5	34.70	**890**	59	5	42.54
r101	173	34	5	23.33	**182**	34	5	24.13	**182**	35	5	31.87
r102	425	48	5	39.50	422	48	5	31.05	**434**	50	5	42.00
r103	**526**	53	5	35.59	522	51	5	36.27	522	52	5	49.11
r104	593	56	5	30.27	594	56	5	31.02	**601**	56	5	53.96
r105	**376**	44	5	25.80	375	43	5	22.62	374	43	5	36.31
r106	**518**	49	5	37.48	509	49	5	36.66	514	49	5	43.32
r107	544	52	5	30.36	540	54	5	37.68	**548**	53	5	42.25
r108	592	54	5	34.78	589	55	5	28.93	**603**	57	5	48.86
r109	492	50	5	31.57	**494**	50	5	31.37	491	50	5	39.01
r110	**521**	51	5	28.26	515	50	5	27.42	517	52	5	34.80
r111	**571**	55	5	26.38	553	54	5	26.33	560	54	5	38.51
r112	608	54	5	30.38	**613**	55	5	33.58	610	56	5	49.31
rc101	**458**	45	5	32.79	**458**	45	5	30.74	**458**	45	5	34.72
rc102	549	46	5	29.43	**563**	48	5	24.58	557	47	5	33.64
rc103	614	50	5	29.55	**616**	49	5	29.92	612	49	5	30.55
rc104	726	53	5	25.76	726	53	5	41.92	**727**	54	5	47.00
rc105	520	46	5	36.97	**526**	47	5	26.23	519	48	5	25.75
rc106	546	50	5	29.81	**549**	51	5	33.61	527	47	5	38.15
rc107	596	49	5	26.96	596	49	5	24.74	**610**	50	5	42.21
rc108	**703**	53	5	33.41	694	53	5	30.01	**716**	53	5	47.45
c201	**1470**	96	3	42.71	1460	95	3	32.70	1410	100	4	26.81
c202	**1450**	94	3	31.78	1440	93	3	34.55	1410	100	4	30.39
c203	1410	100	4	26.51	**1430**	92	3	32.88	1310	100	5	22.34
c204	**1410**	100	4	19.81	**1410**	100	4	20.75	**1410**	100	4	23.34
c205	**1480**	97	3	34.89	1470	96	3	35.44	1410	100	4	22.71
c206	1460	95	3	29.66	**1470**	96	3	29.64	1410	100	4	21.87
c207	1430	92	3	26.01	1460	95	3	32.29	1410	100	4	31.18
c208	**1490**	98	3	31.88	1450	94	3	28.31	1410	100	4	26.38
r201	**1138**	97	3	50.62	1119	95	3	44.49	1058	100	4	34.11
r202	1152	98	3	39.37	**1158**	100	3	33.53	1058	100	4	22.07
r203	**1158**	100	3	31.89	**1158**	100	3	34.35	1058	100	4	20.78
r204	1151	99	3	20.16	**1155**	98	3	20.89	1058	100	4	19.86
r205	**1158**	100	3	40.17	**1158**	100	3	35.28	1058	100	4	22.92
r206	1158	100	3	33.70	**1175**	81	2	31.52	1058	100	4	20.16
r207	**1158**	100	3	27.72	**1158**	100	3	27.65	**1158**	100	3	20.54
r208	**1158**	100	3	18.92	1058	100	4	18.20	1058	100	4	19.37
r209	**1158**	100	3	28.21	**1158**	100	3	34.06	1058	100	4	22.91
r210	**1158**	100	3	31.54	**1158**	100	3	29.26	1058	100	4	20.65
r211	**1158**	100	3	23.25	1124	92	3	20.86	1058	100	4	24.31
rc201	**1368**	92	3	48.57	1355	90	3	44.74	1324	100	4	25.90
rc202	**1414**	99	3	35.25	1324	100	4	30.93	1324	100	4	24.55
rc203	1324	100	4	24.74	1324	100	4	25.44	**1424**	100	3	20.49
rc204	**1424**	100	3	25.46	1324	100	4	24.40	1324	100	4	19.43
rc205	**1385**	95	3	42.46	1384	93	3	33.87	1324	100	4	23.60
rc206	**1421**	99	3	37.52	1411	96	3	32.25	1324	100	4	20.38
rc207	**1412**	97	3	29.25	1324	100	4	28.94	1324	100	4	20.29
rc208	1422	99	3	28.09	**1424**	100	3	25.46	1324	100	4	22.24
#BKS	**27**				25				21			
Gap	**0.795**				1.346				3.083			
Time(s)	31.06				30.57				31.62			

Bold numbers indicate the best solution values found by MSA, SA, or ABC.

speed, memory size, operating system, compiler, computer program, and precision. The proposed MSA algorithm takes no more than 4.0 s to solve small problems, which the Gurobi solver requires much longer to solve. The overall improvement rate (*IR*) is used to measure the degree of improvement. *IR* is computed as $(TNP^{h}_{\max} - TNP^{MILP}_{\max})/TNP^{MILP}_{\max} \times 100\%$, where $TNP^{MILP}_{\max}$ and $TNP^{h}_{\max}$ are the total net profit obtained using MILP and heuristic *h*, respectively. The maximum and average *IR* of MSA over MILP is 17.422% and 1.431%, respectively. In summary, the proposed MSA algorithm obtained better solutions in less computation time than the Gurobi solver.

5.4. Comparing results obtained by MSA, SA, and ABC

Computational experiments were carried out on a large problem set to compare the performance of the MSA with that of the SA and ABC

Table 6
Computational results for problems with 5 mandatory visits and a constant path cost of 125.

Instance	MSA				SA				ABC			
	Best	#visited	#path	time(s)	Best	#visited	#path	time(s)	Best	#visited	#path	time(s)
c101	**565**	51	5	29.56	555	50	5	32.01	**565**	51	5	32.37
c102	**695**	57	5	33.34	685	55	5	26.86	**705**	58	5	40.20
c103	735	56	5	26.82	735	56	5	25.37	**745**	57	5	28.70
c104	765	58	5	34.39	765	58	5	30.00	**775**	59	5	31.73
c105	595	50	5	29.74	595	50	5	26.78	**605**	52	5	40.14
c106	615	52	5	21.80	615	51	5	24.11	**625**	52	5	35.53
c107	655	54	5	27.74	645	54	5	28.74	**655**	54	5	26.98
c108	665	55	5	30.22	665	55	5	32.76	**675**	56	5	38.87
c109	735	57	5	29.25	745	58	5	32.11	**765**	59	5	39.18
r101	**60**	28	4	20.23	59	29	4	17.07	**60**	28	4	35.38
r102	**310**	49	5	30.47	293	47	5	33.31	309	49	5	37.10
r103	373	51	5	32.36	383	52	5	24.04	**392**	51	5	37.15
r104	**468**	55	5	26.87	445	54	5	29.93	463	55	5	43.28
r105	**250**	45	5	26.82	222	36	4	22.44	245	43	5	51.61
r106	376	48	5	34.38	381	49	5	20.77	377	49	5	30.44
r107	**425**	53	5	33.69	408	51	5	29.12	415	53	5	41.46
r108	474	56	5	35.56	**482**	56	5	25.66	477	56	5	34.35
r109	**371**	50	5	28.36	330	40	4	24.06	365	51	5	49.91
r110	**392**	50	5	25.41	364	49	5	25.68	**392**	53	5	41.38
r111	419	54	5	25.17	436	54	5	31.09	433	53	5	47.21
r112	489	56	5	33.61	487	55	5	24.97	**498**	56	5	39.47
rc101	313	43	5	27.73	329	44	5	23.47	**330**	45	5	30.39
rc102	**439**	49	5	27.14	418	46	5	30.22	436	48	5	37.18
rc103	503	50	5	28.00	**507**	51	5	27.45	501	50	5	52.08
rc104	**605**	53	5	31.51	601	53	5	29.00	589	55	5	46.43
rc105	**403**	48	5	25.20	391	45	5	24.03	373	44	5	45.18
rc106	**415**	48	5	34.49	411	48	5	27.16	409	47	5	49.04
rc107	461	47	5	24.57	**469**	49	5	23.95	454	48	5	30.19
rc108	553	53	5	27.72	**581**	55	5	34.95	578	52	5	48.48
c201	1375	94	3	36.84	**1385**	95	3	47.87	1310	100	4	25.41
c202	**1395**	96	3	42.53	1385	95	3	48.21	1310	100	4	24.77
c203	**1345**	91	3	30.78	1325	89	3	29.78	1310	100	4	31.67
c204	1355	92	3	23.02	**1375**	94	3	23.62	1310	100	4	22.15
c205	**1395**	96	3	35.50	**1395**	96	3	40.85	1310	100	4	22.59
c206	1395	96	3	41.40	**1405**	97	3	40.69	1310	100	4	21.92
c207	1385	95	3	40.24	**1395**	96	3	37.55	1310	100	4	22.80
c208	**1415**	98	3	33.58	1375	94	3	31.54	1310	100	4	21.61
r201	1052	95	3	53.56	**1054**	94	3	44.95	1025	92	3	28.20
r202	**1077**	98	3	45.64	**1077**	98	3	38.27	**1077**	98	3	28.15
r203	**1083**	100	3	36.68	**1083**	100	3	36.73	**1083**	100	3	21.01
r204	**1083**	100	3	22.48	**1083**	100	3	25.45	958	100	4	19.66
r205	**1089**	82	2	37.75	1083	100	3	36.83	958	100	4	24.20
r206	**1161**	90	2	41.92	1083	100	3	37.34	958	100	4	19.69
r207	**1083**	100	3	24.32	**1083**	100	3	30.18	**1083**	100	3	19.55
r208	**1083**	100	3	23.73	**1083**	100	3	22.51	958	100	4	19.18
r209	**1083**	100	3	32.72	**1083**	100	3	37.02	**1083**	100	3	23.19
r210	**1083**	100	3	34.46	**1083**	100	3	36.19	958	100	4	20.26
r211	**1083**	100	3	25.03	958	100	4	21.89	958	100	4	19.75
rc201	**1303**	94	3	55.66	1281	90	3	45.09	1301	94	3	37.81
rc202	**1335**	98	3	35.37	1331	98	3	40.14	1328	97	3	26.02
rc203	**1349**	100	3	28.59	**1349**	100	3	38.38	**1349**	100	3	24.89
rc204	**1349**	100	3	22.80	1299	94	3	23.44	1224	100	4	19.35
rc205	**1329**	96	3	51.03	1312	95	3	38.95	1324	97	3	42.11
rc206	1341	98	3	28.93	**1346**	99	3	36.83	1224	100	4	23.35
rc207	**1424**	100	4	28.79	**1424**	100	4	30.88	**1424**	100	4	20.07
rc208	**1424**	100	4	24.72	**1424**	100	4	26.70	**1424**	100	4	19.27
#BKS	**34**				23				21			
Gap	**0.765**				1.903				3.041			
Time(s)	31.79				30.98				31.97			

Bold numbers indicate the best solution values found by MSA, SA, or ABC.

algorithms. The SA is the MSA with $P_{size} = 1$. To have a fair comparison between MSA and SA, the value of $N_{non\text{-}improving}$ is increased to 17 in SA so that the total average computational time for MSA and SA are about the same. Given the computational complexity of TOPTW-MV, a high-quality solution to a large problem is typically not readily obtainable. Therefore, for each benchmark instance in the large problem set, the *IR* values of the MSA, SA, and ABC algorithms were calculated using the best solution that was found by either one of the algorithms, according to the following equation.

$$RPD_h = \frac{TNP^b_{\max} - TNP^h_{\max}}{TNP^b_{\max}} \times 100\%$$

where $TNP^h_{\max}$ and $TNP^b_{\max}$ are the total net profit that was obtained by heuristic h and the best found solution, respectively.

The best solutions obtained by MSA, SA, and ABC for each problem

with five mandatory visits are shown in Tables 4–6. The constant cost of a path for problems in Tables 4, 5 and 6 is 75, 100, and 125, respectively. In these tables, Instance column displays the name of the problem instance, while the numbers in the Best, #visited, #path, and time (s) columns are the total net profit, number of visited customers, number of paths in the best solution obtained among 5 runs, and the average running time per run in seconds. It can be found that the number of best known solutions obtained by MSA is greater than that of SA and ABC in Tables 4–6 (33, 27, and 34 for MSA; 31, 25, and 23 for SA; 25, 21, and 21 for ABC). Moreover, the performance of MSA is better than that of SA and ABC in terms of the average percentage gap to the best solution values for each of these three sets of test instances. It can also be seen that, if the constant cost of a path increases, the number of paths used decreases. This may be due to that the cost of a path cannot be compensated by the net profit gained by serving more non-mandatory visits.

The performance of each algorithm deteriorates as the constant cost of a path increases. As observed earlier, fewer paths are used when the constant cost of a path increases. This means that, on average, each path will visit more customers, which in turn increases the complexity and difficulty of finding optimal paths. Nevertheless, it seems that the rate of deterioration for MSA is much slower than that of SA and ABC, either from the perspective of the number of best solutions found or the average percentage gap to the best solution values (0.364%, 0.795%, and 0.765% for MSA; 0.706%, 1.346%, and 1.903% for SA; 1.550%, 3.083%, and 3.041% for ABC). It is worth mentioning that the average gap between the average solution value and the best solution value from 5 runs of MSA is only 2.14%. This shows the robustness of the proposed MSA heuristic.

We further applied statistical tests to compare the performance of MSA, SA, and ABC algorithms. To determine whether the MSA algorithm outperformed the SA and ABC algorithms, one-sided paired t-tests in terms of *Min. RPD*, *Ave. RPD*, and *Max. RPD* were conducted. The best, average, and worst total net profit of the solutions to each test problem, based on five runs, obtained using MSA and SA, were used to compute *RPD* values, which were denoted as *Min. RPD*, *Ave. RPD*, and *Max. RPD*. The statistical results that were obtained in Table 7 revealed that, at a confidence level of $\alpha = 0.05$, the proposed MSA algorithm significantly outperformed SA and ABC algorithms in terms of *Min. RPD* and *Ave. RPD,* for overall problems. These results indicate that the multi-start strategy in MSA significantly improves the performance of SA on solving TOPTW-MV.

Table 7
Paired-test on *Min. RPD*, *Ave. RPD*, and *Max. RPD*.

	Problem Set	RPD	Difference	Dof	p-value
MSA vs. SA	$\gamma = 75$	*Min. RPD*	−0.3416	55	0.0316*
		Ave. RPD	−0.0830	55	0.3094
		Max. RPD	−0.2685	55	0.1616
	$\gamma = 100$	*Min. RPD*	−0.5511	55	0.0355*
		Ave. RPD	−0.2058	55	0.1636
		Max. RPD	−0.0402	55	0.4655
	$\gamma = 125$	*Min. RPD*	−1.4079	55	0.0074*
		Ave. RPD	−0.6329	55	0.0320*
		Max. RPD	−06068	55	0.1408*
	Overall	*Min. RPD*	−1.6769	167	0.0002*
		Ave. RPD	−0.3072	167	0.0160*
		Max. RPD	−0.3052	167	0.1168
MSA vs. ABC	$\gamma = 75$	*Min. RPD*	−1.1862	55	0.0003*
		Ave. RPD	−1.4084	55	0.0000*
		Max. RPD	−1.6345	55	0.0001*
	$\gamma = 100$	*Min. RPD*	−2.2881	55	0.0001*
		Ave. RPD	−1.4327	55	0.0003*
		Max. RPD	−0.9264	55	0.0495
	$\gamma = 125$	*Min. RPD*	−2.2752	55	0.0004*
		Ave. RPD	−0.9424	55	0.0684
		Max. RPD	1.0617	55	0.0783
	Overall	*Min. RPD*	−1.9165	167	0.0000*
		Ave. RPD	−1.2612	167	0.0000*
		Max. RPD	−0.4997	167	0.738

* Significant difference exists between MSA and SA/ABC.

6. Conclusion and future research

This work concerns the TOPTW-MV, which is a challenging extension of TOPTW. To reduce the gap between theory and industrial practice, this work develops an MILP model of TOPTW-MV and a multi-start simulated annealing algorithm, which incorporates a multi-start strategy for solving the problem. The proposed MSA algorithm can significantly reduce search effort while maximizing total net profit in solving TOPTW-MV. Experimental results reveal that MSA yields higher-quality solutions to TOPTW-MV than SA and ABC do.

TOPTW-MV has many practical applications. Related problems, such as those with various different objective functions, can be investigated in the future. Future research may also attempt to apply other meta-heuristics or to hybridize other algorithms to solve TOPTW-MV.

Acknowledgements

The first author is grateful to the Ministry of Science and Technology of the Republic of China (Taiwan) and the Chang Gung Memorial Hospital, Linkou for financially supporting this research under grants MOST 105-2410-H-182-009-MY2 and CMRPD3G0011, respectively. The work of the second author is partially supported by the Ministry of Science and Technology of the Republic of China (Taiwan) under grant MOST 103-2221-E-011-062-MY3. This support is gratefully acknowledged.

References

Archetti, C., Hertz, A., & Speranza, M. G. (2007). Metaheuristics for the team orienteering problem. *Journal of Heuristics, 13*(1), 49–76.
Campos, V., Marti, R., Sanchez-Oro, J., & Duarte, A. (2014). GRASP with path relinking for the orienteering problem. *Journal of the Operational Research Society, 65*(12), 1800–1813.
Chao, I., Golden, B. L., & Wasil, E. A. (1996). The team orienteering problem. *European Journal of Operational Research, 88*(3), 464–474.
Chekuri, C., Korula, N., & Pal, M. (2012). Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms* 8(3), Article No. 23.
Cura, T. (2014). An artificial bee colony algorithm approach for the team orienteering problem with time windows. *Computers & Industrial Engineering, 74*, 270–290.
Dang, D.-C., Guibadj, R. N., & Moukrim, A. (2013). An effective PSO-inspired algorithm for the team orienteering problem. *European Journal of Operational Research, 229*(2), 332–344.
Duque, D., Lozano, L., & Medaglia, A. L. (2015). Solving the orienteering problem with time windows via the pulse framework. *Computers & Operations Research, 54*, 168–176.
El-Hajj, R., Dang, D. C., & Moukrim, A. (2016). Solving the team orienteering problem with cutting planes. *Computers & Operations Research, 74*, 21–30.
Fischetti, M., Gonzalez, J. J. S., & Toth, P. (1998). Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing, 10*(2), 133–148.
Gedik, R., Kirac, E., Milburn, A. B., & Rainwater, C. (2017). A constraint programming approach for the team orienteering problem with time windows. *Computers & Industrial Engineering, 107*, 178–195.
Gendreau, M., Laporte, G., & Semet, F. (1998). A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks, 32*(4), 263–273.
Golden, B. L., Levy, L., & Vohra, R. (1987). The orienteering problem. *Naval Research Logistics, 34*(3), 307–318.
Gunawan, A., Lau, H. C., & Lu, K. (2015). An iterated local search algorithm for solving the orienteering problem with time windows. *Evolutionary Computation in Combinatorial Optimization, Evocop, 2015*(9026), 61–73.
Gunawan, A., Lau, H. C., & Vansteenwegen, P. (2016). Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research, 255*(2), 315–332.
Hu, Q., & Lim, A. (2014). An iterative three-component heuristic for the team orienteering problem with time windows. *European Journal of Operational Research, 232*(2), 276–286.
Kantor, M. G., & Rosenwein, M. B. (1992). The orienteering problem with time windows. *Journal of the Operational Research Society, 43*(6), 629–635.
Ke, L. J., Zhai, L. P., Li, J., & Chan, F. T. S. (2016). Pareto mimic algorithm: An approach to the team orienteering problem. *Omega-International Journal of Management Science,*

61, 155–166.
Keshtkaran, M., Ziarati, K., Bettinelli, A., & Vigo, D. (2016). Enhanced exact solution methods for the team orienteering problem. *International Journal of Production Research, 54*(2), 591–601.
Kim, B.-I., Li, H., & Johnson, A. L. (2013). An augmented large neighborhood search method for solving the team orienteering problem. *Expert Systems with Applications, 40*(8), 3065–3072.
Labadie, N., Mansini, R., Melechovský, J., & Wolfler Calvo, R. (2012). The team orienteering problem with time windows: An LP-based granular variable neighborhood search. *European Journal of Operational Research, 220*(1), 15–27.
Labadie, N., Melechovský, J., & Wolfler Calvo, R. (2010). Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. *Journal of Heuristics, 17*(6), 729–753.
Leifer, A. C., & Rosenwein, M. B. (1994). Strong linear-programming relaxations for the orienteering problem. *European Journal of Operational Research, 73*(3), 517–523.
Lin, S. W. (2013). Solving the team orienteering problem using effective multi-start simulated annealing. *Applied Soft Computing, 13*(2), 1064–1073.
Lin, S.-W., & Yu, V. F. (2012). A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research, 217*(1), 94–107.
Lin, S. W., & Yu, V. F. (2015). A simulated annealing heuristic for the multiconstraint team orienteering problem with multiple time windows. *Applied Soft Computing, 37*, 632–642.
Mann, M., Zion, B., Rubinstein, D., Linker, R., & Shmulevich, I. (2016). The orienteering problem with time windows applied to robotic melon harvesting. *Journal of Optimization Theory and Applications, 168*(1), 246–267.
Muthuswamy, S., & Lam, S. (2011). Discrete particle swarm optimization for the orienteering problem. *International Journal of Industrial Engineering: Theory, Applications and Practice, 18*(2), 92–102.
Palomo-Martínez, P. J., Salazar-Aguilar, M. A., Laporte, G., & Langevin, A. (2017). A hybrid variable neighborhood search for the orienteering problem with mandatory visits and exclusionary constraints. *Computers & Operations Research, 78*, 408–419.
Righini, G., & Salani, M. (2009). Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & Operations Research, 36*(4), 1191–1203.
Salazar-Aguilar, M. A., Langevin, A., & Laporte, G. (2014). The multi-district team orienteering problem. *Computers & Operations Research, 41*, 76–82.
Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., & Van Oudheusden, D. (2010). A path relinking approach for the team orienteering problem. *Computers & Operations Research, 37*(11), 1853–1859.
Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., & Van Oudheusden, D. (2013). The multiconstraint team orienteering problem with multiple time windows. *Transportation Science, 47*(1), 53–63.
Tae, H., & Kim, B. I. (2015). A branch-and-price approach for the team orienteering problem with time windows. *International Journal of Industrial Engineering: Theory, Applications and Practice, 22*(2), 243–251.
Tang, H., & Miller-Hooks, E. (2005). A TABU search heuristic for the team orienteering problem. *Computers & Operations Research, 32*(6), 1379–1407.
Tricoire, F., Romauch, M., Doerner, K. F., & Hartl, R. F. (2010). Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research, 37*(2), 351–367.
Vansteenwegen, P., Souffriau, W., & Van Oudheusden, D. (2011). The orienteering problem: A survey. *European Journal of Operational Research, 209*(1), 1–10.
Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., & Van Oudheusden, D. (2009a). A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research, 196*(1), 118–127.
Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., & Van Oudheusden, D. (2009b). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research, 36*(12), 3281–3290.
Wang, Q. W., Sun, X. Y., Golden, B. L., & Jia, J. Y. (1995). Using artificial neural networks to solve the orienteering problem. *Annals of Operations Research, 61*, 111–120.
Yu, V. F., Jewpanya, P., Ting, C.-J., & Redi, A. A. N. P. (2017). Two-level particle swarm optimization for the multi-modal team orienteering problem with time windows. *Applied Soft Computing, 61*, 1022–1040.