12-2019

Selective discrete particle swarm optimization for the team orienteering problem with time windows and partial scores

Vincent F. YU

Perwira A. A. N. REDI

Parida JEWPANYA

Aldy GUNAWAN
Singapore Management University, aldygunawan@smu.edu.sg

Citation

YU, Vincent F.; REDI, Perwira A. A. N.; JEWPANYA, Parida; and GUNAWAN, Aldy. Selective discrete particle swarm optimization for the team orienteering problem with time windows and partial scores. (2019). *Computers and Industrial Engineering*. 138, 1-13.
Available at: https://ink.library.smu.edu.sg/sis_research/4469

Selective discrete particle swarm optimization for the team orienteering problem with time windows and partial scores

Vincent F. Yua, A.A.N. Perwira Redia, Parida Jewpanyaa, Aldy Gunawan

Abstract

This paper introduces the Team Orienteering Problem with Time Windows and Partial Scores (TOPTW-PS), which is an extension of the Team Orienteering Problem with Time Windows (TOPTW). In the context of the TOPTW-PS, each node is associated with a set of scores with respect to a set of attributes. The objective of TOPTW-PS is to find a set of routes that maximizes the total score collected from a subset of attributes when visiting the nodes subject to the time budget and the time window at each visited node. We develop a mathematical model and propose a discrete version of the Particle Swarm Optimization (PSO), namely, the Selective-Discrete PSO (S-DPSO), to solve TOPTW-PS. The proposed S-DPSO uses four different movement schemes to move a particle from its current position. The best movement scheme is selected to determine the next position of the particle. To evaluate the performance of the proposed S-DPSO algorithm, we first test S-DPSO on two variants of Orienteering Problem, namely, Team Orienteering Problem (TOP) and TOPTW. Experimental results show that S-DPSO performs well in solving benchmark instances of TOP and TOPTW. In general, S-DPSO is comparable to the state-of-the-art algorithms for these problems. We also apply the S-DPSO to solve 168 newly generated TOPTW-PS instances and conclude that the proposed S-DPSO can produce high-quality TOPTW-PS solutions.

Keywords Team orienteering problem, Time window, Partial score, Discrete particle swarm optimization

1. Introduction

The Orienteering Problem (OP) was first introduced by Tsiligirides (1984). In OP, a set of vertices associated with a location and a score is given. The time required to travel between each pair of locations is known in advance. OP aims to find a route that visits a subset of vertices to maximize the total collected score (Vansteenwegen, Souffriau, Vanden Berghe, & Van Oudheusden, 2009a). Therefore, in OP, determining the selection of vertices and shortest path between selected vertices are important factors. OP can be considered as a combination of the Knapsack Problem and the Travelling Salesman Problem (TSP). In the classical TSP, the objective is to determine a path that visits all vertices with the smallest travel distance. The Prize Collecting TSP (PCTSP) is a variant of TSP in which all vertices do not need to be visited. PCTSP considers a reward/prize for every visited vertex and a penalty to every vertex that is not visited (Balas, 1989). Meanwhile, a TSP variant with the objective of finding a circuit that minimizes travel cost minus collected profit is known as the Profitable Tour Problem (PTP) (Dell'Amico, Maffioli, & Värbrand, 1995). The three TSP variants (OP, PCTSP and PTP) are considered as the generic forms of TSP with profits (Feillet, Dejax, & Gendreau, 2005). A well-known OP extension is the Team OP (TOP), which incorporates multiple routes. Another extension that is widely considered in OP and TOP is the TOP with Time Windows (TOPTW), which incorporates the time window constraints. Vansteenwegen et al. (2011b) comprehensively surveyed about OP, TOP, and TOPTW. Gunawan, Lau, and Vansteenwegen (2016) extended the survey by including the latest variants of the OP, such as the proposed solution approaches and the most recent applications of the OP.

OP and its variants have attracted attention because of the numerous real-life applications (Gedik et al., 2017, Yu et al., 2017a). In this paper, we focus on the application of TOPTW to the Tourist Trip Design Problem (TTDP). TTDP is a route-planning problem for tourists that are interested in visiting multiple points of interest (POIs). Typically, most tourists visit as many POIs as possible during their limited time (Vansteenwegen et al., 2009a). However, given the time limitations, they might not be able to visit all POIs. Thus, tourists generally select those locations with high scores or valuable POIs (Vansteenwegen et al., 2011a). TTDP aims to help tourists in determining tours when visiting POIs in a customized trip.

In visiting each POI, the score is assumed to come from one type of activity. Nevertheless, in reality, upon a visit to a POI, several activities can be selected. For example, when visiting a beach, tourists can experience several leisure activities, such as taking photo of good views and doing some water sports. Each activity requires a certain period of time and provides a different valuable score. In this case, the problem is further complicated because the tourists should decide where they should visit and what activities they should attend. Therefore, the present paper aims to present an extension of TOPTW, namely, TOPTW and Partial Scores (TOPTW-PS). TOPTW-PS maximizes the total collected scores based on the visited POIs and the combination of selected activities.

The proposed TOPTW-PS is explained with the following example. Tourists plan to visit a certain area with several POIs. At each POI, several activities can be selected; each activity is associated with a score and the amount of time needed to finish it. TOPTW-PS aims to maximize the total collected score with several constraint limitations, such as the time budget and time windows. When considering the nature of the TOPTW-PS, its application is not only limited to the trip design problem. TOPTW-PS can also be applied in humanitarian logistic, where a volunteer selects the route to visit and the necessary activities to be conducted in the shelter. The application of routing technicians or sales representatives to serve customers with several optional requested services is also one possible application of the TOPTW-PS.

TOPTW-PS is a new and challenging problem. It is an extension of TOPTW which belongs to the class of non-polynomial-hard (NP-hard) problem (Golden, Levy, & Vohra, 1987). Hence, TOPTW-PS is also NP-hard. Hence, TOPTW-PS is a NP-hard problem. To deal with this complex problem, we develop the discrete version of the well-known Particle Swarm Optimization (PSO), namely, the Selective Discrete Particle Swarm Optimization (S-DPSO). In summary, the contribution of this paper is threefold:

- We present a new variant of the TOPTW, namely, TOPTW-PS, in which each vertex is associated with a set of scores with respect to a set of activities.
- We generate a new set of benchmark instances for TOPTW-PS by extending the characteristics of the benchmark TOPTW instances.
- We propose a discrete version of the well-known PSO, namely, S-DPSO, to solve the TOPTW-PS. S-DPSO introduces four different movement schemes: (1) following its own position, (2) moving toward its personal best position, (3) moving toward the global best position, and (4) moving toward the combination of the three above-mentioned schemes. The best movement scheme in term of an objective function is selected for each particle to move to the next position. The ability of the proposed S-DPSO to solve the TOP and the TOPTW is also tested.

The remainder of this paper is organized as follows. Section 2 reviews the literature related to the proposed problem and the solution approach. Section 3 presents the problem definition, including the mathematical model. Section 4 describes the original discrete PSO concept. Section 5 demonstrates the proposed S-DPSO for solving the TOPTW-PS, including the solution representation and movement schemes. Section 6 provides the experimental results. Finally, Section 7 concludes this study and suggests some future research directions.

2. Related work

The topics on OP, TOP, and their applications have been studied for many years (Schilde et al., 2009, Tsiligirides, 1984, Vansteenwegen and Van Oudheusden, 2007, Wang et al., 2008). The original OP comes from the orienteering game (Chao, Golden, & Wasil, 1996), which is generally played in a forested area. OP aims to determine which route maximizes the collecting point or score (Chao et al., 1996). Vansteenwegen et al. (2011b) reviewed the OP, TOP, and its extensions. Tang and Miller-Hooks (2005) provided a detailed review on the published solution approaches for OP and TOP.

OP and TOP with time windows are also introduced. In OPTW and TOPTW, the time window is considered a constraint. Among practical situations, motivating the inclusion of time windows include routing problem cases where each customer or location should be visited within a predetermined time interval. For example, in TTDP, some attractive places may have a specific time they are open. Therefore, tourists should plan their trip to visit during the opening period of each place.

Abbaspour and Samadzadegan (2011) introduced the tour planning extension of OPTW. The tour-planning problem is modeled in the context of the time-dependent OPTW. The time-independent tour planning in a large urban area considers multimodal transportations. This problem determines the sequences for visiting the attractive points during a specific period via several modes of transportation system. To solve the problem, an evolutionary strategy based on a genetic algorithm is proposed. Garcia, Vansteenwegen, Arbelaitz, Souffriau, and Linaza (2013) introduced a personalized electronic tourist guide to help tourists decide which places to visit and obtain transportation information. They modeled the tourist planning problem by integrating public transportation as the time-dependent TOPTW; they solved this problem using the algorithm based on Vansteenwegen, Souffriau, Vanden Berghe, & Van Oudheusden (2009b). Souffriau, Vansteenwegen, Vanden Berghe, and Van Oudheusden (2013) proposed the multiconstraint TOP with multiple time windows; in this problem, each vertex is extended with additional knapsack constraints, which limit the selection of vertices, together with the time windows and the time budget. In the tourist application, these additional constraints will involve budget limitations for entrance fees for each day (e.g., a maximum number of museums to visit on the first day). The hybrid solution mechanism, which integrates iterated local search and a greedy randomized adaptive search procedure (GRASP), was proposed to solve this problem.

In terms of solution methods, several algorithms, including the exact method (Boussier et al., 2007, Fischetti et al., 1998) and heuristic methods, have been used to solve OP and its extensions. The basic OP model uses the exact method to determine the optimal solution; however, this method is considerably difficult and usually requires a long computational time. Therefore, heuristic and evolutionary computing methods have been applied to determine a near-optimal solution in a reasonable amount of time. Vansteenwegen et al. (2011b) provided a detailed published solution approach for OP and its extensions. Metaheuristic approaches, such as tabu search (Tang & Miller-Hooks, 2005), ant colony optimization (ACO) (Ke et al., 2008, Montemanni and Gambardella, 2009), variable neighborhood search (VNS) (Campbell et al., 2011, Tricoire et al., 2010), iterated local search heuristic (ILS) (Gunawan et al., 2015, Vansteenwegen et al., 2009b), simulated annealing (SA) (Lin & Yu, 2012), artificial bee colony (ABC) (Cura, 2014), PSO (Dang et al., 2013, Muthuswamy and Lam, 2011, Yu et al., 2017a), and the hybrid approach (Labadie, Melechovský, & Wolfler Calvo, 2010), deliver efficient results for some specified problems.

3. Problem definition and model formulation

In TOPTW-PS, the fixed number of tours or paths is given. A set of attractive vertices associated with multiple activities and time window is introduced. Each activity in the vertex is associated with a score and a service time. The TOPTW-PS aims to determine the paths that maximize the total collected score from visiting vertices and attending some activities without violating their time windows. For the mathematical model formulation, we define the following:

- A set N={N0∪Nc} is a set of all vertices. N0 is a set of vertex 0 that is designated as the starting and end vertices, that is, N0={0}. Nc is a set of vertices that represents the attractive vertices (Nc={1,2,3,...,|N|}). *P* is a set of paths, that is, P={1,2,3,...,|P|}. *A* is a set of activities provided in each vertex, that is, A={1,2,3,...,|A|}. All vertices are assumed to have the same set of activities.
- A partial score Si,a is associated with vertex i∈Nc and activity a∈A. wi,a represents the service time associated with vertex *i* in activity *a*.
- A travel time ti,jis the time for traveling from vertex *i* to *j* (i,j∈N).
- Each vertex i∈Nccan be visited at most once in the time interval [Oi,Ei]. Moreover, not all vertices can be visited because of the limited given time T_{max}.

Decision variablesai,a=1Ifanactivityaisselectedbyavisittovertexi0Otherwisexi,j,p=1Ifinpathp,avisittovertexiisfollowedbyavisittovertexj0Otherwiseyi,p=1Ifinpathp,avertexiisvisited0Otherwise

vi	=	Total visiting time of a visit to vertex *i*
si,p	=	Starting time of a visit to vertex *i* in path *p*

Objective function(1)max∑p∈P∑i∈Nc∑a∈ASi,a·ai,a·yi,p

Constraints(2)∑∀p∈P∑∀j∈Ncx0,j,p=∑∀p∈P∑∀i∈Ncxi,0,p=|P|(3)∑∀i∈N,i≠kxi,k,p=∑∀j∈N,k≠jxk,j,p=yk,p∀k∈Nc,∀p∈P(4)∑∀p∈Pyi,p⩽1∀i∈Nc(5)∑∀a∈Aai,a⩽A∀i∈Nc(6)∑∀a∈Awi,a·ai,a⩽vi∀i∈Nc(7)si,p+vi+ti,j-sj,p⩽M(1-xi,j,p)∀i∈N,∀j∈Nc,∀p∈P(8)Oi⩽si,p∀i∈Nc,∀p∈P(9)si,p⩽Ei∀i∈Nc,∀p∈P(10)si,p+ti,0·xi,0,p⩽Tmax∀i∈Nc,∀p∈P(11)sj,p⩾t0,j,v·yj,p∀j∈Nc,∀p∈P(12)xi,j,p∈0,1∀i,j∈N,p∈P(13)ai,a∈0,1∀i∈N,p∈P(14)yi,p∈0,1∀i,j∈N,p∈PObjective function (1) maximizes the total collected score. Constraint (2) guarantees that each path starts and ends in Vertex 0. Constraint (3) ensures the path connectivity through flow conservation equalities. Constraint (4) ensures that each vertex is visited at most one. Constraint (5) ensures that most |*A*| activities can be selected in each vertex. Constraint (6) calculates the total visiting time of each vertex. Constraint (7) determines the timeline of each path. Constraints (8), (9) restrict the start of the service to the time window. The route is limited by Tmax (Constraint (10)). In addition, the visiting time of the first visited vertex must be longer than the travel time of arc (0*, j*) (Constraint (11)). Finally, constraints (12), (13), (14) are the integrality constraints for the decision variables.

4. Particle swarm optimization (PSO)

PSO was proposed by Kennedy and Eberhart (1995). The standard PSO algorithm is initially designed to solve continuous optimization problems. PSO imitates the physical movements of individuals/particles in the swarm as a searching method. Each particle represents a solution to the problem and its move directly relates to changes in the solution. The particles move through the search space seeking the global optimum. However, there is no guarantee of reaching the global optimum. Furthermore, particle moves are influenced by the information such as its own experience and globally shared information from other particles in the swarm.

We denote the most suitable solution of particle *k* at iteration *t* (the personal best position) as pk-bestt and the global best solution taken from all particles at iteration *t* (the particle swarm's global best position) as gbestt. At iteration *t*, particle *k* updates its next position xkt+1 by using Eqs. (15), (16) (Muthuswamy & Lam, 2011):(15)vkt+1=w·vkt+c1·r1(pk-bestt-xkt)+c2·r2(gbestt-xkt);(16)xkt+1=xkt+vkt+1;where vkt and xkt represent the velocity and the position of particle *k* at iteration *t*, respectively; *w* is a constant value that controls the impact of the previous velocity; c_1 is the cognitive parameter for remembering the most suitable particle position that has been reached; c_2 is the social parameter controlling the communication among particles to converge toward the global best position; and r_1 and r_2 are uniformly distributed random variables in [0,1].

There are two versions of PSO which are commonly used to solve discrete optimization problems such as VRP. The first version is the real-valued version of PSO. In this version, the particle position is represented as a string of arrays that consists of real numbers. The solution to the problem is generated through a decoding procedure which normally involves a mechanism to transform real values in the solution to integers or binary numbers. Examples of real-valued PSO can be found in Ai and Kachitvichyanukul (2009). Moreover, Yu, Jewpanya, and Kachitvichyanukul (2016) used the same concept to solve the transshipment problem. The discrete-valued PSO represents the solution by a string of integers or binary numbers. In the discrete-valued PSO, the solution can be directly obtained by decoding the particle position. An illustration between these versions is illustrated in Fig. 1.

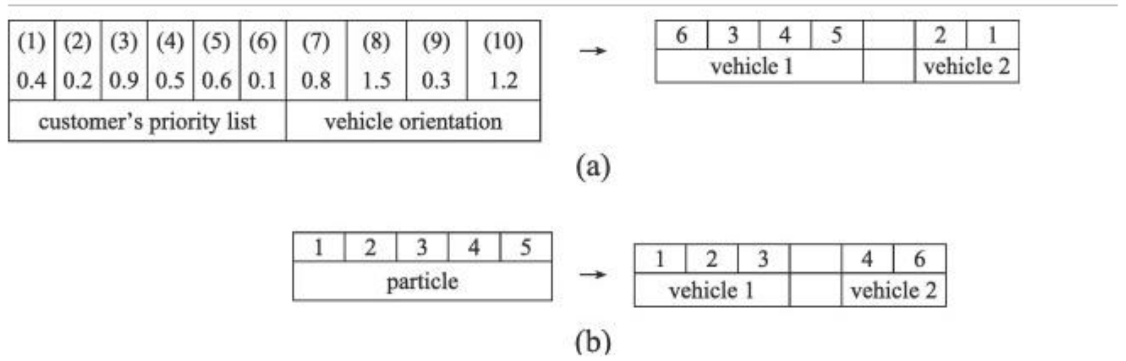


Fig. 1. Illustration of (a) real-valued PSO by Ai and Kachitvichyanukul (2009) and (b) discrete-valued PSO by Goksal et al. (2013).

The example of real-valued PSO can be seen in Yu et al. (2017a). This study applied a variant of PSO called 2L-GLNPSO for solving the Multi-Modal Team Orienteering Problem with Time Windows (MM-TOPTW). This approach is the closest implementation of PSO to our proposed S-DPSO. However, because the decision variables of MM-TOPTW are discrete, 2L-GLNPSO needs a decoding process to convert the real-valued solution representation to the discrete-valued one. Instead of using real-valued solution representation similar to 2L-GLNPSO, our proposed S-DPSO are based on the discrete-valued PSO (DPSO).

The decoding process is a process to convert the encoded solution from a form of real-valued solution representation to a discrete-valued solution representation. At the end of the process, the goal is to simplify the calculation of the objective function value. In general, the setup of a real-valued solution representation is similar for most problems. In the solution representation, elements of the solution length and the value range are defined. Takes example of the 2L-GLNPSO implemented for solving MM-TOPTW by Yu et al. (2017a), the solution representation is divided into two levels of real-valued solution. The first level is to represent the priority list of vertices that shows the sequence of vertices being visited. It is decoded by using a sorting function of the real-valued number in the solution, and then taking the index as the new representation of the solution. The second level represents the type of transportation being selected. This solution is decoded by assigning the real-valued solution representation to the representative integer value defined by a function explained in the paper. Then, this process takes information regarding how the objective function value is calculated to obtain the objective value. However, our proposed algorithm is extended from DPSO which does not need decoding process since the form of solution representation is already discrete. Some of the optimization problems solved using DPSO include TSP (Pang, Wang, Zhou, Dong, Liu, & Zhang, 2004), VRP (Goksal et al., 2013, Gong et al., 2012), scheduling problem (Kashan and Karimi, 2009, Pan et al., 2008, Tseng and Liao, 2008), and OP (Dang et al., 2013, Muthuswamy and Lam, 2011).

PSO has been reported to be an algorithm of fast convergence and easy implementation. Therefore, in the present research, we propose the S-DPSO for TOPTW-PS. S-DPSO is a DPSO extension with several mechanisms to improve the particle movement. The mechanism that provides the best solution is used for particle movement. Therefore, in this research, we propose the S-DPSO with several mechanisms to improve the particle movement. The mechanism that provides the most suitable solution is used for particle movement. The use of S-DPSO in solving TOPTW-PS is explained in the following section.

5. Proposed S-DPSO algorithm for TOPTW-PS

This section introduces the use of S-DPSO in solving TOPTW-PS. S-DPSO is a discrete version of PSO that implements a discrete permutation-based solution. The structure of this discrete-valued solution representation and the path construction procedure are explained in Section 5.1. The parameters used in this proposed algorithm is described in Section 5.2. In the process of moving a particle from its initial position to another position, S-DPSO uses four mechanisms to generate four candidate solutions which are explained in Section 5.3. Each solution consists of the position and the associated objective value. The mechanism on how S-DPSO solves TOPTW-PS is explained in 5.4 Local search, 5.5 S-DPSO procedure.

5.1. Solution representation and path construction procedure

The solution representation for TOPTW-PS is derived from the solution representation for TOPTW by Lin and Yu (2012). It consists of two parts. The first part is represented by a permutation of |Nc|vertices, followed by |*P*|-1 zeros for separating paths. Therefore, the total dimension of the first part is |Nc|+|P|-1. The second part of the solution representation is used to represent the selected activity when visiting each vertex, and it consists of |Nc| dimensions. The value in each dimension is an integer number, which is generated randomly from 1 to (2|A|-1). Each integer number in [1, (2|A|-1)] represents one specific type of selection from all possible (2|A|-1) types. Each integer number is subsequently encoded to a binary variable ac_i, that is, i∈A, to obtain the selected activities. If the activity *i* is selected, then ac_i is 1; otherwise, it is 0. The integers in dimension 1 to |Nc| represent the activity selection of visiting vertex 1 to vertex |Nc|. This approach can help reduce the dimension of the solution representation and enable multiple activity selections.

Table 1 provides a TOPTW-PS instance with 25 vertices. Each vertex shows three activities (|A|=3). In this example, the vertex's coordinate (*X, Y*), a service time (W_a), and a partial score (S_a) associated with activity a∈A, earliest time (O_i), and latest time (E_i) are also listed. When we consider two paths (*p* = 2), the total number of dimensions is 51 (total dimensions consist of decisions for activity (|Nc|) and visited vertices (|Nc|+|P|-1)).

Table 1. TOPTW-PS instance with 25 locations and one, two, or three activities at each location.

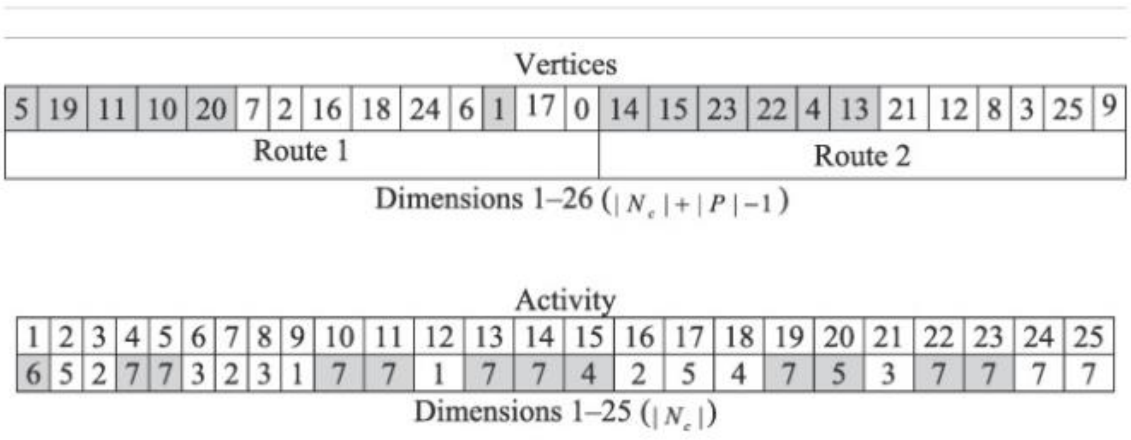
No.	***X***	***Y***	W_1	W_2	W_3	***O***	***E***	S_1	S_2	S_3
0	40	50	0	0	0	0	1236	0	0	0
1	45	68	0	5	5	912	967	0	45	45
2	45	70	3	21	6	825	870	9	63	18
3	42	66	5	2	3	65	146	45	18	27
4	42	68	3	6	1	727	782	27	54	9
5	42	65	8	0	2	15	67	72	0	18
6	40	69	2	14	4	621	702	9	63	18
7	40	66	14	2	4	170	225	63	9	18
8	38	68	6	0	14	255	324	27	0	63
9	38	70	0	3	7	534	605	0	27	63
10	35	66	1	1	8	357	410	9	9	72
11	35	69	9	0	1	448	505	81	0	9
12	25	85	2	0	18	652	721	9	0	81
13	22	75	0	9	21	30	92	0	27	63
14	22	85	5	2	3	567	620	45	18	27
15	20	80	24	4	12	384	429	54	9	27
16	20	85	4	20	16	475	528	9	45	36
17	18	75	8	4	8	99	148	36	18	36
18	15	75	0	12	8	179	254	0	54	36
19	15	80	7	2	1	278	345	63	18	9
20	30	50	5	1	4	10	73	45	9	36

No.	X	Y	W_1	W_2	W_3	O	E	S_1	S_2	S_3
21	30	52	16	0	4	914	965	72	0	18
22	28	52	18	0	2	812	883	81	0	9
23	28	55	5	0	5	732	777	45	0	45
24	25	50	4	1	5	65	144	36	9	45
25	25	52	32	0	8	169	224	72	0	18

In TOPTW-PS, visitors visiting a vertex can select more than one activity. Table 2 illustrates the activity selection from three activities ($|A| = 3$) at each vertex. In this case, there are seven possible activity combinations denoted by the integers 1–7. Each number is encoded into binary variables that represent the selected activity, as shown in Table 2. For example, the first dimension of the second part of solution representation is 6 (see Fig. 2). This number represents the activity selection of vertex 1 and it is encoded into binary variables $ac_1 = 0$, $ac_2 = 1$ and $ac_3 = 1$. The interpretation is that when visiting vertex 1, the selected activities are activities 2 and 3, and the total service time and the total score collected at this vertex are 10 and 90, respectively.

Table 2. Possible activity combinations for a vertex with three activities.

Activity combination	ac_i		
	$i = 3$	$i = 2$	$i = 1$
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1



Vertices

5	19	11	10	20	7	2	16	18	24	6	1	17	0	14	15	23	22	4	13	21	12	8	3	25	9
Route 1														Route 2											

Dimensions 1–26 ($|N_c|+|P|-1$)

Activity

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
6	5	2	7	7	3	2	3	1	7	7	1	7	7	4	2	5	4	7	5	3	7	7	7	7

Dimensions 1–25 ($|N_c|$)

Fig. 2. Example of solution representation for a problem with 25 vertices and 2 paths.

Fig. 2 illustrates an example of the solution representation of the data displayed in Table 1. The complete visual illustration of solution to this instance is shown in Fig. 3. In this example, the first path starts by visiting the vertex 5 and subsequently vertices 19, 11, 10, 20, and 1, and it ends with a zero. The vertices 7, 2, 16, 18, 24, 6, and 17 are excluded from the first path because visiting these vertices violates the time window, maximum travel time, or budget constraint. In the first path, when visitors visit the first four vertices (vertices 5, 19, 11, and 10), the integer value in the 5th, 19th, 11th, and 10th dimensions of the second representation is 7. Therefore, all activities are selected. For the last two visited vertices (vertices 1 and 20), the values of the 1st and 2nd dimensions are 5 and 6, respectively, that is, the visitors only select activities 1 and 3 in vertex 20, and activities 2 and 3 in vertex 1. The procedure is repeated in the second path.

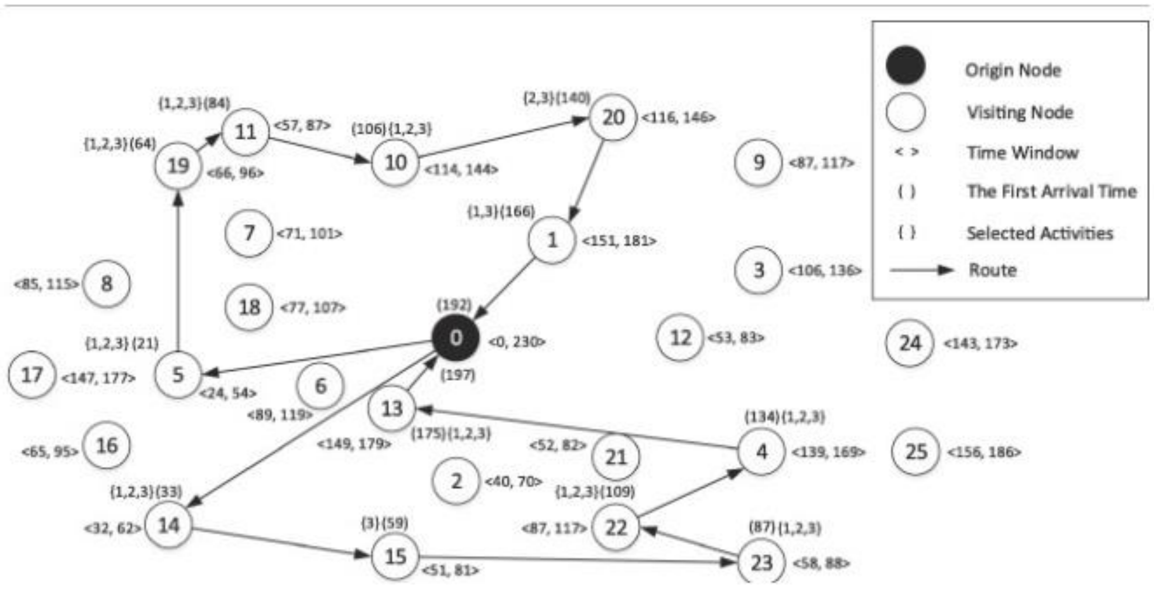


Fig. 3. Visual illustration of the example solution given in Fig. 1.

5.2. Parameters used

The discrete S-DPSO uses six parameters, namely, *I*, *T*, *w*, c_1, c_2, and *L*. *I* denotes the number of particles, and *T* is the number of iterations. The number of particles and iterations are important parameters of PSO because they affect the fitness value and computational time. Furthermore, the increasing size of the population increases computational time, but it might not improve the fitness value. *w* denotes inertia weight, a parameter that controls the search behavior of the swarm. This vector shows the same direction of the current velocity. c_1 and c_2 are acceleration constants of the personal and global best positions, respectively. Each acceleration constant controls the maximum distance that a particle is allowed to move from the current position to each best position. A number of local search improvements (*L*) is also introduced.

5.3. Position updating rule

The position updating rule in S-DPSO is developed based on the discrete domain. S-DPSO allows particles to evaluate each independent move (its current position, its *Pbest* and *Gbest*). Particles may have a chance to select a better position to move than that with moving only toward all of them by comparing several position choices.

In S-DPSO, the position of a particle *k* at iteration *t* is updated by four possible constructive updating mechanisms: (1) following its own position (Xkt), (2) going toward its *Pbest* (Pit), (3) going toward *Gbest* (Pgt), and (4) going toward its own position, namely, *Pbest* and *Gbest*. These mechanisms generate four candidate solutions. Each solution consists of the position and the associated objective value. The v1kt,v2kt,v3kt,andv4ktare the updated positions generated by mechanisms (1), (2), (3), and (4), respectively. Correspondingly, the objective values associated with these positions aref(v1kt),f(v2kt),f(v3kt),andf(v4kt), respectively. The most suitable candidate solution in terms of the objective value is selected and updated as the next position of the particle *k*. The process of generating a new position for a selected particle is depicted in the following equations:(17)v1kt+1=(w⨂Xkt)⨁Rkt;(18)v2kt+1=(c1⨂Pkt)⨁Xkt;(19)v3kt+1=(c2⨂Pgt)⨁Xkt;(20)v4kt+1=(c2⨂Pgt)⨁((c1⨂Pkt)⨁((w⨂Xkt)⨁Rkt));and(21)Xkt+1=argmaxf(v),v∈{v1kt+1,v2kt+1,v3kt+1,v4kt+1}The definitions of the operators used in Eqs. (17), (18), (19), (20), (21) are as follows. Eq. (17) is the position construction function for (1)'s constructive updating rule. In this rule, the new position is formulated following the previous position (Xkt) with *w* acceleration. In Eq. (17), Rktdenotes the duplication of Xkt. This approach can update the particle using its own position. Eqs. (18), (19) formulate the position following the personal best (Pkt) and the global best (Pgt) with c1 and c2, respectively. Eq. (20) is utilized to update the position toward own, personal best, and global best positions. The definitions of the operators used in Eqs. (17), (18), (19), (20) are as follows.

Eq. (21) is used to select the position with the maximum objective value to be the new position for particle *k* in the next iteration (*t* + 1).
The multiply operator(⨂) is used to select the dimensions of customers fromXkt,Pkt, andPgt. For the first |Nc|+|P|-1 dimensions, the process starts with determining the number of selected dimensions of Xkt,Pkt, and Pgt, which are denoted as sw,s1, and s2, respectively. These numbers are calculated from acceleration numbers *w*, c_1, and c_2 multiplied by a number of particle dimensions, which represent the vertices, that is, |Nc|+|P|-1. Afterward, the process randomly selectssw, s1, and s2dimensions from Xkt,Pkt, and Pgt, respectively. The numbers of selected dimensions are calculated as follows.(22)sw=(|Nc|+|P|-1)·w;(23)s1=(|Nc|+|P|-1)·c1;(24)s2=(|Nc|+|P|-1)·c2;where · denotes the smallest integer that is larger than or equal to the enclosed number. For the second part of the solution representation, the dimensions of the particle are maintained.

For example, in Fig. 4, five vertices (|Nc|=5) are considered with three types of activity (|A|=3) in one path (|P|=1). Thus, the particle represents the vertices with five dimensions. The example of generating new position in Eq. (17) starts at determining swby using Eq. (22). We assume that the acceleration number *w* is 0.5. Therefore, the number of selected dimensions is 3, that is, sw=5∗0.5=3. The process then randomly selects three dimensions from Xkt, as shown in Fig. 4.

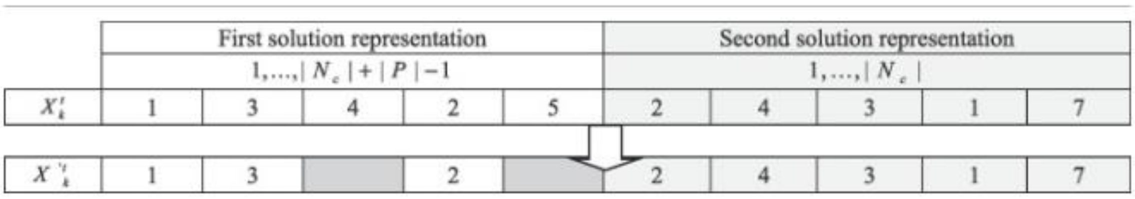


Fig. 4. Example of multiply operator (⨂).

The add operator (⨁) is used to construct the solution after the multiply operator. In the first |Nc|+|P|-1 dimensions, the non-duplicated dimensions from the added particle are inserted into the empty dimension of multiply operator result. For the activity selection dimension, the process randomly selects the dimension from the operated particle or the dimension of the multiply operator result.

Fig. 5 displays the add operator process between the multiply operator result (X’kt) and the added particle (Rkt). The Rkt dimension is added into the sequence of X’kt. The Rkt dimension that is duplicated with X’kt is not considered. Hence, in this section, 5 and 4 are added toX’kt.

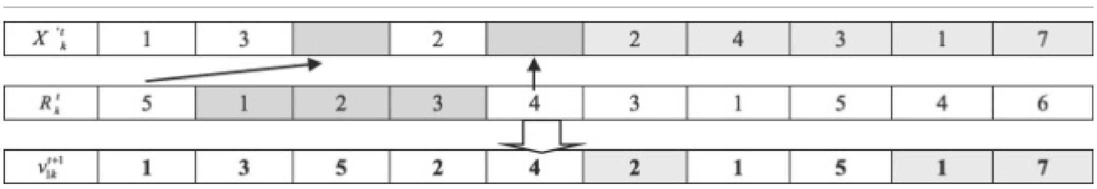


Fig. 5. Example of add operator (⊕).

5.4. Local search

Local search is implemented to determine the best activity selection of each vertex whenever a new *Gbest* is found*.* The local search is executed *L* times at each iteration in the second part of the solution representation, which consists of N_c dimensions. The local search steps are as follows.

Algorithm 1

Step 1:	Select dimension *i* in *Gbest* randomly, i∈1,...,Nc
Step 2:	Randomly change the value of dimension *i* from the possible activity
Step 3:	Evaluate the feasibility of the modified activity and the score improvement
Step 4:	If feasibility is maintained, and the path score is improved, then keep the modified route; otherwise, return the route to the condition prior to step 1

To ensure the feasibility of the solution, only new feasible solutions are accepted. Moreover, as mentioned in Step 4, if a new feasible solution improves the objective value, it is used as the result of the procedure. The feasibility checking mechanism examines if the newly selected activity is available for the selected vertices.

5.5. S-DPSO procedure

In the proposed algorithm, Xkt is a position vector of a particle *k* at iteration *t*. At the beginning, the iteration number (*T*) is zero, and an initial solution Xkt for all particle *k* is randomly generated. The *Pbest* and *Gbest* are updated.

At each iteration, a new position Xkt+1 is updated from the four types of updating rules, which are known as the particle velocities. The next particle position selected is that with the most objective value. The *Pbest* of each particle is compared with the new Xkt+1; if the current Xkt+1 obtained a better solution than *Pbest*, the *Pbest* is replaced by Xkt+1. Otherwise, the *Pbest* result is maintained. When all particle positions are updated, the *Gbest* can be obtained. In addition, in the case of S-DPSO algorithm, the *Gbest* found at each iteration is improved by the local search in Algorithm 1. The algorithm is terminated when the number of iteration is satisfied. A S-DPSO procedure of solving TOPTW-PS is given in Algorithm 2.

Algorithm 2

```
begin
t←0;
for all k∈Kdo
Xkt←Generate a particle at random;
Pkt←Xkt;
end for
Pgt←\{Plt|l=ergmax∀k{f(Xkt)\} \} ;
(t<T) do
for all k∈K do
v1kt+1←Update thekthparticle velocity vector using (17)
v2kt+1←Update thekthparticle velocity vector using (18)
v3kt+1←Update thekthparticle velocity vector using (19)
v4kt+1←Update thekthparticle velocity vector using (20)
Xkt+1←Update thekthparticle position vector using (21)
if f(Xkt+1)>f(Pkt)
Pkt+1←Xkt+1
else
Pkt+1←Pkt
end if
end for
if f(Pgt)<max∀k{f(Pkt+1)}>f(Pkt)
```

```
$P_g^{t+1} \leftarrow \{P_l^{t+1} | l = \arg\max_{\forall k}\{f(P_k^{t+1})\}\}$;
do Algorithm 1.
end if
$t \leftarrow t+1$
end while
end
```

6. Computational study

The proposed S-DPSO algorithm is written in C++. All experiments are performed on a PC with a 3.4-GHz processor and 20-GB RAM under the Windows 7 operating system. Each experiment of TOPTW-PS is performed 10 times. The experiment for each TOPTW instance is performed five times. The best and average results are presented. S-DPSO is first tested on TOP and TOPTW benchmark instances. Results are compared with those obtained by state-of-the-art approaches. Moreover, three sets of TOPTW-PS instances are generated: small scale (25 vertices), medium scale (50 vertices), and large–scale problems (100 vertices). Each set consists of 56 instances. Results of small-scale and medium-scale instances are compared with the CPLEX results. For large-scale problem, S-DPSO results are compared with those obtained by the S-DPSO, which only implements the position updating rule (4) in the searching process, namely, S-DPSO-(4). Finally, a statistical analysis is conducted to provide a comprehensive discussion describing the statistical test of the algorithm performance.

6.1. Test problems

6.1.1. TOPTW instances

The benchmark TOPTW instances were designed by Montemanni and Gambardella (2009). The instance used in Montemanni and Gambardella (2009) is based on their own OPTW instances and the OPTW instances of Righini and Salani (2008). Righini and Salani (2008) designed test instances for OPTW using 29 Solomon (1987) datasets of VRP with time windows (c10*, r10*, and rc10*), which are denoted as Solomon 100, and 10 datasets of multidepot VRP of Cordeau, Gendreau, and Laporte (1997)(pr1–pr10), which are denoted as Cordeau 1–10. Montemanni and Gambardella (2009) added 27 additional instances based on 10 Solomon (1987) (c20*, r20*, and rc20*), which are denoted as Solomon 200, and 10 instances based on Cordeau et al. (1997) (pr11–pr20), which are denoted as Cordeau 11–20. These benchmark instances can be downloaded at http://www.mech.kuleuven.be/en/cib/op.

6.1.2. TOPTW-PS instances

TOPTW-PS instances were designed based on Montemanni and Gambardella (2009) which derived them from the Solomon (1987) datasets (c10*, c20*, r10*, r20*, rc10*, and rc20*). The Solomon (1987) datasets consist of six different classes, namely, R1, R2, C1, C2, RC1, and RC2. These instances are named according to the distribution of the vertex location in the Cartesian coordinates. The vertex locations in classes R1 and R2 are randomly generated based on a uniform distribution. Classes C1 and C2 locations are clustered. Finally, classes RC1 and RC2 contain semi clustered vertex locations that combined both clustered and randomly distributed locations. In each instance, each vertex is associated with a vertex location, a time window (starting and closing time), a service time, and demand. The three categories consist of instances with 100, 50, and 25 vertices without the depot. Each category consists of total 56 instances.

The vertex position and time windows in TOPTW-PS instances are the same as those in the original TOPTW instances. In TOPTW-PS, each vertex *i* has several activities *a* that are associated with a partial score $S_{i,a}$ and a service time $w_{i,a}$. The number of activities at each vertex *i*, denoted as $|A_i|$, is randomly generated from {1, …, *n*}, where *n* is the maximum number of activities at each vertex. The score and service time of vertex *i* in the original TOPTW instances are Si' and STi', respectively. The new score of each activity *a* at vertex *i* is generated by Si,a=U(1,Si'), where ∑i=1|Nc|Si,a=Si'. Using the same approach, the new service time of each activity *a* at vertex *i* is generated by STi,a=U(1,STi'), where ∑i=1|Nc|STi,a=STi'. When the new test sets are designed in this manner, this particular high-quality TOPTW solution is also a feasible high-quality solution for the new TOPTW-PS problem. Therefore, a considerably good solution to compare with is known. These TOPTW-PS instances can be downloaded at http://web.ntust.edu.tw/~vincent/op/.

6.2. Parameter setting

To obtain a good performance of the proposed S-DPSO in terms of solution quality and computing time, we conduct the experiment to determine the best parameter setting for each problem. The experiment consists of two steps: first, we use one factor at time (OFAT) in which the levels for each parameter, and second are changed sequentially by fixing the value of other parameter. Table 3 shows combinations of parameter values being evaluated.

Table 3. Parameter settings of S-DPSO for solving TOP, TOPTW, and TOPTW-PS.

Problem	Candidate parameter	Selected parameter
TOPTW	I = {5, 10, 30}; T = {500,8000,1000}; w = {0.3,0.6,0.9}; C_1 = {0.3,0.6,0.9}; C_2 = {0.3,0.6,0.9}; and L = {500, 1000, 2000}	I = 30; T = 800; w = 0.9; C_1 = 0.6; C_2 = 0.6; and L = 1000
TOPTWPS		I = 10; T = 1000; w = 0.9; C_1 = 0.6; C_2 = 0.6; and L = 500

6.3. Statistical analysis

The algorithm's performance is statistically evaluated using the equivalent nonparametric test of ANOVA, which is the Kruskal–Wallis test statistics. Prior to the Kruskal–Wallis test statistic, the mean rank of algorithm results relative to each other is calculated. The small mean rank value shows that the algorithm result is good. To support the significance of other results, the Kruskal–Wallis test is necessary. The Kruskal–Wallis procedure assumes no normal distribution for the residuals, and it could be utilized to compare independent instances, which may present different sample sizes. These statistical procedures are similar to those conducted by Yu et al. (2017b). This approach specifies which method or group of methods significantly differs from each other by conducting post-test using the Mann–Whitney *U* test.

6.4. Computational results and analysis

6.4.1. Comparison results on TOPTW instances

The S-DPSO is tested on available benchmark TOPTW instances. Results are compared with the results obtained from ACO (Montemanni & Gambardella, 2009), ILS (Vansteenwegen et al., 2009b), VNS (Tricoire et al., 2010), SSA (Lin & Yu, 2012), GRASP-ELS (Labadie, Melechovský, & Calvo, 2011), GVNS (Labadie, Mansini, Melechovský, & Wolfler Calvo, 2012), I3CH (Hu & Lim, 2014), and ABC (Cura, 2014). We could not obtain the original source codes of the algorithms. Thus, we prefer to present the original results of those algorithms, which are also shown in the study of Cura (2014). Notably, the ILS and SSA are executed only once; the ACO, GRASP-ELS, GVNS, I3CH, and ABC are executed five times; and the VNS is executed for 10 times. The S-DPSO is executed five times follows the ABC implementation. According to the preliminary testing, the previously mentioned parameter values exhibit the best performance within a reasonable computational time.

Table 4 shows that the gap of average result for S-DPSO ranges from 0.00% to 2.72% for test problems with m = 1, 2, 3, 4. The average CPU ranges from 0.18 s to 207.83 s. In term of the average gap, S-DPSO is better than GVNS, VNS, SSA, GRASP-ELS, ILS, ACS and ABC. However, to obtain the solutions it is slower than some of those algorithms, GVNS, GRASP-ELS, ILS and ABC. Furthermore, I3CH can obtain the best average gap. The statistical analysis of these algorithms results are conducted further.

Table 4. Results for TOPTW benchmark instances.

Instance set	***N***	**GVNS**		**VNS**		**SSA**		**GRASP-ELS**		**ILS**		**ACS**		**ABC**		**I3CH**		**S-DPSO**	
		Gap avg. (%)	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**
p = *m* = 1																			
Solomon 100	29	2.46	66.54	0.07	85.39	0.05	22.32	0.20	9.01	1.94	0.24	0.10	200.13	0.46	4.42	0.69	26.71	0.19	20.59
Solomon 200	27	2.54	75.53	0.54	857.80	0.50	44.67	1.14	16.52	2.53	1.67	0.54	857.80	0.97	21.35	1.34	132.14	0.69	63.43
Cordeau 1–10	10	1.44	12.37	0.92	822.05	0.80	112.21	1.28	5.03	4.56	1.75	1.03	1626.63	1.30	78.54	1.05	109.01	0.33	98.41
Cordeau 11–20	10	3.51	24.22	2.61	1045.94	2.97	162.40	2.61	7.90	8.88	1.98	11.22	887.65	2.78	103.67	4.28	130.23	2.72	85.42
p = *m* = 2																			
Solomon 100	29	1.64	73.88	0.96	68.78	0.04	34.52	1.28	26.60	1.85	0.89	0.63	1278.64	0.32	13.00	0.47	69.31	0.85	60.37
Solomon 200	27	1.16	19.79	0.70	813.69	0.65	76.87	0.38	20.91	2.80	2.60	2.88	2222.72	0.85	30.65	0.47	463.80	1.26	80.19

Instance set	*N*	GVNS		VNS		SSA		GRASP-ELS		ILS		ACS		ABC		I3CH		S-DPSO	
		Gap avg. (%)	CPU (s)	Gap avg. (%)	CPU (s)	Gap avg. (%)	CPU (s)	Gap avg. (%)	CPU (s)	Gap avg. (%)	CPU (s)	Gap avg. (%)	CPU (s)	Gap avg. (%)	CPU (s)	Gap avg. (%)	CPU (s)	Gap avg. (%)	CPU (s)
Cordeau 1–10	10	1.18	39.09	3.30	524.83	1.86	173.93	1.61	19.46	5.64	4.76	2.98	1889.66	2.45	149.69	1.04	247.06	1.66	142.09
Cordeau 11–20	10	1.52	82.44	2.98	618.79	3.24	201.63	2.92	28.77	7.24	5.21	5.53	2384.81	2.99	221.84	2.70	304.61	2.41	142.46
p = *m* = 3																			
Solomon 100	29	1.83	91.09	1.18	68.86	0.40	45.95	1.56	35.04	2.36	1.46	1.41	1421.73	0.59	22.55	0.20	135.85	0.10	93.71
Solomon 200	27	0.24	7.32	0.11	322.34	0.45	52.26	0.06	3.50	1.10	1.71	0.82	1344.96	0.32	12.49	0.02	89.24	0.36	16.69
Cordeau 1–10	10	0.86	85.90	3.37	473.21	2.07	197.01	1.71	40.55	6.33	9.24	3.74	2163.80	2.64	228.93	0.36	424.00	0.75	178.34
Cordeau 11–20	10	1.39	150.73	2.95	517.47	3.42	251.83	2.86	42.95	8.82	9.69	6.23	2228.16	3.84	247.64	1.11	496.95	1.80	181.32
p = *m* = 4																			
Solomon 100	29	1.85	86.59	1.56	66.85	0.48	58.32	1.53	39.93	3.12	2.42	1.67	1523.01	0.95	29.54	0.16	199.54	0.32	114.25

Instance set	***N***	**GVNS**		**VNS**		**SSA**		**GRASP-ELS**		**ILS**		**ACS**		**ABC**		**I3CH**		**S-DPSO**	
		Gap avg. (%)	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**	**Gap avg. (%)**	**CPU (s)**
Solomon 200	27	0.00	0.53	0.00	141.21	0.00	40.45	0.00	0.02	0.00	1.02	0.02	245.42	0.00	0.53	0.00	0.17	0.04	0.18
Cordeau 1–10	10	1.30	127.33	3.25	403.17	1.90	255.57	2.20	45.75	6.77	14.07	3.27	2447.70	3.17	233.28	0.36	566.51	1.61	205.14
Cordeau 11–20	10	2.45	232.64	3.14	408.00	3.60	283.98	3.07	65.33	8.13	13.74	6.00	2583.50	3.49	267.58	0.45	728.62	1.48	207.83
Average		**1.58**	**73.50**	**1.73**	**452.40**	**1.40**	**125.87**	**1.53**	**25.46**	**4.50**	**4.53**	**3.00**	**1581.65**	**1.69**	**104.11**	**0.92**	**257.74**	**1.04**	**105.65**

As shown in Table 5, the mean ranks for GVNS, VNS, SSA, GRASP-ELS, ILS, ACS, ABC, I3CH, and S-DPSO are 73.09, 79.06, 62.63, 69.25, 110.94, 86.97, 72.44, 36.69, and 67.44, respectively. The Kruskal–Wallis test results in Table 6 shows a significant difference from at least one algorithm with at least one of other algorithms tested for TOPTW results.

Table 5. Mean ranks of TOPTW algorithms.

Problem	**Algorithm**	**N**	**Mean rank**
TOPTW	GVNS	16	74.16
	VNS	16	73.28
	SSA	16	63.78
	GRASP-ELS	16	70.16
	ILS	16	111.16

Problem	Algorithm	N	Mean rank
	ACS	16	87.41
	ABC	16	72.72
	I3CH	16	46.28
	S-DPSO	16	53.56

Table 6. Kruskal–Wallis test statistics for results of TOPTW algorithms (GVNS, VNS, SSA, GRASP-ELS, ILS, ACS, ABC, I3CH).

	Average percentage gap
Chi-square	28.543
Df	8
Asymp. Sig.	0.000
Critical value (left-tailed, right-tailed)	(2.73, 15.51)

We conduct the Mann–Whitney *U* test statistic on TOPTW results. The process starts with division of the algorithms into three groups (A1, A2, and A3), as shown in Table 7. This grouping is based on a preliminary Kruskal–Wallis test. Subsequently, the algorithm results in each group are tested by the Kruskal–Wallis test. In this case, only group A1 is performed because it consists of more than one member (see Table 8). Results in Table 9 show no significantly different result in the member of group A1. Finally, the results of each group are tested using the Mann–Whitney *U* test with 95% confidence interval (α is 0.05 and critical value is −1.64). Result in Table 9 shows that the average gap between each group is significantly different. Based on this experiment, we can conclude that I3CH is the most effective among the algorithms for TOPTW. Our proposed S-DPSO algorithm can generate results as good as those of GVNS, VNS, SSA, GRASP-ELS, ACS, and ABC.

Table 7. Grouping of GVNS, VNS, SSA, GRASP-ELS, ILS, ACS, ABC, I3CH, and S-DPSO algorithm for Mann–Whitney *U* test statistics.

Group	Member
A1	GVNS, VNS, SSA, GRASP-ELS, ACS, ABC, and S-DPSO
A2	ILS

Group	Member
A3	I3CH

Table 8. Kruskal–Wallis test statistics for A1 (GVNS, VNS, SSA, GRASP-ELS, ACS, ABC, and S-DPSO) results.

	Average percentage gap
Chi-square	6.443
Df	6
Asymp. Sig.	0.375
Critical value (left-tailed, right-tailed)	(1.64, 12.59)

Table 9. Mann–Whitney *U* test statistics for TOPTW benchmark results.

Algorithm 1	Algorithm 2	Mann-Whitney U	Z	Asymp. Sig. (two-tailed)
A1	A2	376.000	−3.747	0.000
A1	A3	575.000	−2.313	0.021
A2	A3	29.500	−3.713	0.000

6.4.2. Comparison results on TOPTW-PS small- and medium-scale instances

The S-DPSO results are compared with those results obtained by CPLEX. The CPLEX solver is terminated after 2 h when the optimal solution is not obtained. We use the same parameter setting for solving these instances. The percentage of gap between the CPLEX and S-DPSO results is calculated as follows.(25)Gap2=CPLEX-S_DPSOCPLEX×100%

Table 10 reports the computational results of small-scale and medium-scale instances. The comparison between CPLEX and S-DPSO results showed that S-DPSO can obtain the optimal solutions. For the medium-scale instances, out of 56 instances, CPLEX obtains the optimal solution for only two instances. The proposed S-DPSO can reach the optimal solutions to those two instances. On average, S-DPSO outperforms the CPLEX by 0.57% and 1.6% for the small and medium scale instances respectively. S-DPSO can also provide the solutions in lower CPU time than that of CPLEX in some instances.

Table 10. Summary of CPLEX and S-DPSO for small and medium size TOPTW-PS instances.

Set	CPLEX		S-DPSO		Gap 2	
	Small (S)	Medium (M)	Small (S)	Medium (M)	S (%)	M (%)
p = m = 1						
Solomon 100	205.52	237.76	210.38	243.93	−2.37%	−2.60%
Solomon 200	429.30	655.93	429.30	674.93	0.00%	−2.90%
p = m = 2						
Solomon 100	342.17	423.21	346.28	436.97	−1.20%	−3.25%
Solomon 200	431.56	825.48	431.56	833.70	0.00%	−1.00%
p = m = 3						
Solomon 100	408.97	570.97	412.93	580.41	−0.97%	−1.65%
Solomon 200	431.56	835.89	431.56	835.96	0.00%	−0.01%
Average	374.84	591.54	377.00	600.98	−0.57%	−1.60%

6.4.3. Comparison results between S-DPSO and S-DPSO-(4) on TOPTW-PS large-scale instances

The large-scale instances are solved using two versions of the S-DPSO, the proposed S-DPSO and the S-DPSO that only implements the position updating rule (4) in the search process, namely, S-DPSO-(4). S-DPSO-(4) uses Eq. (26) to update particle positions. The percentage gap between the average S-DPSO and average S-DPSO-(4) results is calculated as follows.(26)Gap3=(S_DPSO_(4))-(S_DPSO)S_DPSO_(4)×100%The comparison results between S-DPSO and S-DPSO-(4) for the large-scale instances of TOPTW-PS are summarized in Table 11. Results showed that the S-DPSO outperforms the S-DPSO-(4). Moreover, the average computational time of S-DPSO is only slightly slower than that of S-DPSO-(4). The average objective and computational time of S-DPSO-(4) are 914.01 and 219.58 s, respectively. The average objective and computational time of S-DPSO are 982.78 and 255.41 s, respectively. The average percentage gap between the S-DPSO-(4) and S-DPSO objective values is − 8.68%.

Table 11. Summary of S-DPSO-(4) and S-DPSO results for large TOPTW-PS instances.

Set	**S-DPSO-(4)**		**S-DPSO**		**Gap 3**
	Objective	**CPU (s)**	**Objective**	**CPU (s)**	**(%)**
p = m = 1					
Solomon 100	268.07	137.24	298.55	142.67	−11.40
Solomon 200	804.89	138.12	886.85	140.43	−10.58
p = m = 2					
Solomon 100	478.86	193.30	530.00	222.17	−11.58
Solomon 200	1239.96	185.85	1345.78	219.64	−8.86
p = m = 3					
Solomon 100	669.62	233.24	715.48	277.72	−7.53
Solomon 200	1484.52	243.40	1566.52	293.86	−5.87
p = m = 4					
Solomon 100	824.21	319.42	883.03	374.66	−7.23
Solomon 200	1541.96	306.05	1636.00	372.11	−6.39
Average	914.01	219.58	982.78	255.41	−8.68

A more detailed analysis of problem characteristics is shown in Fig. 6. The TOPTW-PS instances characteristics are inherited from TOPTW where they are categorized into clustered (C), random (R), and random-clustered (RC) based on the locations of the vertices. SDPSO outperforms SDPSO-(4) in all categories in terms of objective value. However, it comes with the cost of higher computational time. Both algorithms show an increasing computational time along with the increasing number of days/paths. SDPSO-(4) requires less computational time for most of the instances except those having random characteristics such as 1-R100, 1-R200, 1-RC200, and 3-RC200.

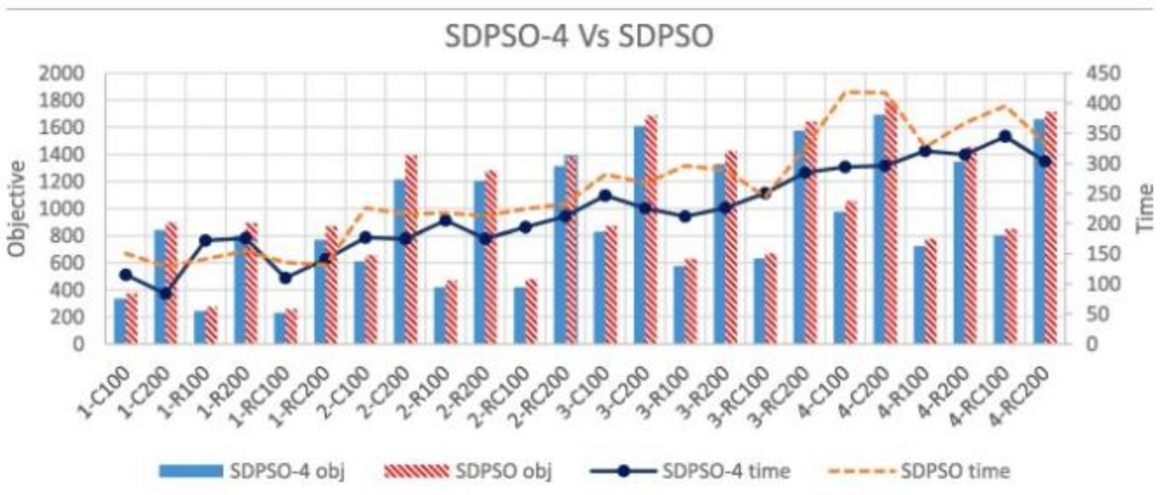


Fig. 6. Illustration of SDPSO-4 and SDPSO results for TOPTW-PS large instances.

We use the S-DPSO and S-DPSO-(4) objective value results to conduct the statistical test analysis for the TOPTW-PS large benchmark instances. The Wilcoxon signed-rank test statistics is conducted with 95% confidence interval (α is 0.05 and critical value is −1.64). Table 12 shows that the average objective values between S-DPSO and S-DPSO-(4) results are significantly different. Therefore, can be interpreted that S-DPSO outperforms S-DPSO-(4).

Table 12. Wilcoxon signed-rank test statistics for S-DPSO and S-DPSO-(4) on TOPTW-PS results.

	Objective value
Z	−12.976
Asymp. Sig.	0.000

6.4.4. Comparative results between S-DPSO and S-DPSO-NoLocal on TOPTW-PS instances

The purpose of adding the local search into S-DPSO is to improve the performance of the algorithm. This section shows the impact of the local search on solving large-scale benchmark instances. An insight of the effectiveness of the local search is shown in Fig. 7. Both algorithms show results with the trend of increasing objectives values and computational times as a result of increasing number of days/paths being considered. This is expected as the more paths considered, the more vertices can be visited, but more computational time is needed to reach the optimal solution. An interesting result is shown in terms of objective value. It shows that the S-DPSO outperforms SDPSO-NoLocal. Although local search requires more computational time, the increased computational time is acceptable considering the improvement in the solutions quality.

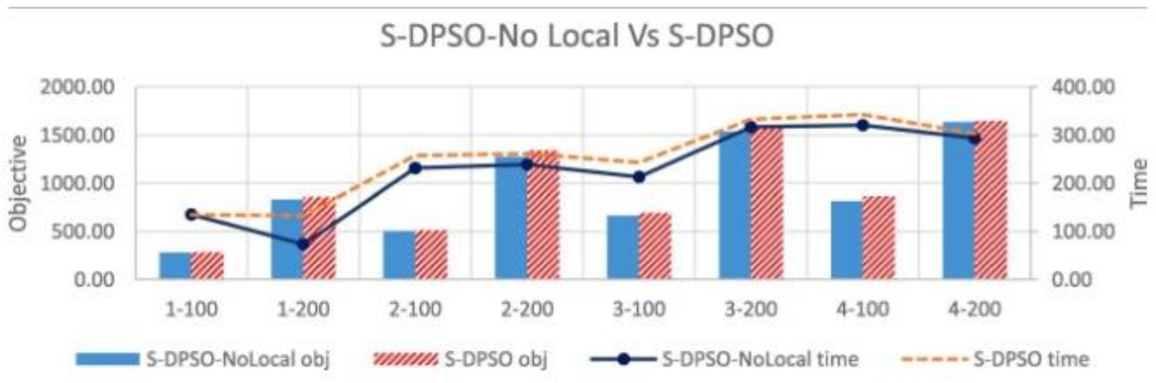


Fig. 7. Illustration of SDPSO-NoLocal and SDPSO results for selected TOPTW-PS large instances.

The results of S-DPSO without local search (S-DPSO-NoLocal) and S-DPSO with local search are summarized in Table 13. The average objective value and computational time of S-DPSO-NoLocal are 943.68 and 227.83 s, respectively, and the average percentage gap between S-DPSO-NoLocal and S-DPSO objective values is −3.68%. The percentage gap between the average S-DPSO and average S-DPSO-NoLocal results is calculated as follows.(27)Gap4=(S_DPSO_NoLocal)-(S_DPSO)S_DPSO_NoLocal×100%

Table 13. Summary of S-DPSO-NoLocal and S-DPSO results for large TOPTW-PS instances.

Set*	**S-DPSO-NoLocal**		**S-DPSO**		**Gap5**
	Objective	**CPU (s)**	**Objective**	**CPU (s)**	**(%)**
p = m = 1					
Solomon 100	281.20	135.21	284.00	135.14	−0.93
Solomon 200	834.20	73.41	857.50	132.81	−2.75
p = m = 2					
Solomon 100	503.40	231.29	514.00	257.28	−1.81
Solomon 200	1275.23	238.80	1339.17	261.52	−4.95
p = m = 3					
Solomon 100	664.13	212.73	694.00	243.15	−4.60
Solomon 200	1538.83	316.83	1576.50	333.26	−2.54

Set*	S-DPSO-NoLocal		S-DPSO		Gap5
	Objective	CPU (s)	Objective	CPU (s)	(%)
p = m = 4					
Solomon 100	815.60	320.66	866.00	342.99	−6.47
Solomon 200	1636.87	293.76	1649.17	302.80	−0.80
Average	943.68	227.84	972.54	251.12	−3.11

*

Conducted only on instances: c101, c109, r101, r112, rc101, rc108, c201, c208, r201, r211, rc201, rc208.

6.4.5. Analysis of TOPTW-PS with randomized scores

The generation procedure for TOPTW-PS instances described in Section 6.1.2 assumes that an activity with larger service time has a higher score. However, this assumption may not be always valid. Moreover, in the TOPTW-PS instances, the proportion of score of each activity is equal to the proportion of the service time of the activity. In this experiment, the dataset introduced earlier is modified by randomly swapping the scores between two activities in each instance for a predetermined number of times. The new dataset is denoted as TOPTW-PS-R instances.

The S-DPSO results for TOPTW-PS and TOPTW-PS-R are shown in Table 14. There is no clear difference between TOPTW-PS and TOPTW-PS-R results. We cannot expect that the TOPTW-PS-R always gives better results than TOPTW-PS does, as shown in Fig. 8. For *m* = 1, TOPTW-PS-R results are better than TOPTW-PS ones. For *m* = 2, 3 and 4, the results of Solomon 100 and Solomon 200 instances show different conclusions. TOPTW-PS gives better results for Solomon 200 instances compared to TOPTW-PS-R. On the other hand, TOPTW-PS-R performs better for Solomon 100. The percentage gap between the average TOPTW-PS and average TOPTW-PS-R results is calculated as follows.(28)Gap5=(TOPTW_PS_R)-(TOPTW_PS)TOPTW_PS_R×100%

Table 14. Summary of S-DPSO results for large TOPTW-PS and TOPTW-PS-RS instances.

Set*_	TOPTW-PS-R		TOPTW-PS		Gap 5
	Objective	**CPU (s)**	**Objective**	**CPU (s)**	**(%)**
p = m = 1					
Solomon 100	325.30	158.00	284.00	135.14	11.66
Solomon 200	861.13	133.32	857.50	132.81	0.30
p = m = 2					
Solomon 100	867.60	233.01	514.00	257.28	24.63
Solomon 200	1013.57	252.25	1339.17	261.52	−57.18
p = m = 3					
Solomon 100	733.17	223.88	694.00	243.15	4.63
Solomon 200	1567.67	316.81	1576.50	333.26	−0.66
p = m = 4					
Solomon 100	879.37	325.81	866.00	342.99	0.75
Solomon 200	1641.73	306.71	1649.17	302.80	−0.48
Average	986.19	243.72	972.54	251.12	−2.04

*

Conducted only on instances: c101, c109, r101, r112, rc101, rc108, c201, c208, r201, r211, rc201, rc208.

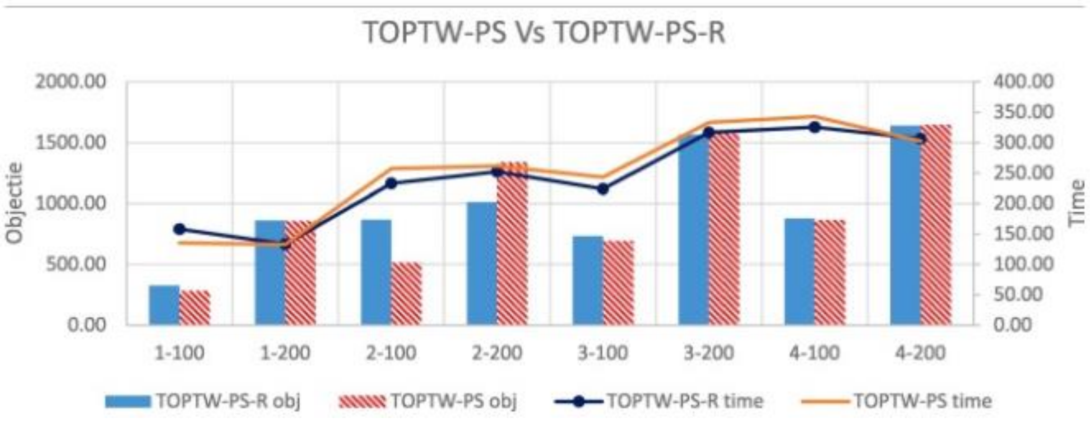


Fig. 8. Illustration of TOPTW-PS-R and TOPTW-PS.

7. Conclusions and further work

In this study, we introduce TOPTW-PS, which is an extension of TOPTW. In TOPTW-PS, each location shows a set of activities. Each activity is associated with a score that is available for visitors to collect. We formulate the mathematical model and propose the S-DPSO to solve the problem. The S-DPSO uses the most suitable one among the four movement schemes for each particle to improve the solution. The proposed S-DPSO is first tested on TOPTW benchmark problems. The computational experiments showed that S-DPSO can produce a high-quality solution for TOPTW, the computational results showed that the proposed S-DPSO can generate solutions as good as those of GVNS, VNS, SSA, GRASP-ELS, ILS, ACS, and ABC. Subsequently, S-DPSO is implemented to solve the TOPTW-PS instances. The computational experiment on TOPTW-PS problem showed that S-DPSO can determine the optimal solution for small-scale instances, and results are similar to those obtained by CPLEX. For large-scale instances, S-DPSO results are significantly better than those of S-DPSO-(4). Overall, S-DPSO results indicated efficiency in solving TOPTW-PS problem. The computational time is reasonable with respect to the size and difficulty of the tested instances. In addition, an analysis of the effect of local search for S-DPSO is conducted. It is shown that the local search improves the S-DPSO algorithm computational results. Moreover, we conduct an experiment to see the difference of TOPTW-PS results if the score and service time is asymmetric.

Future research may consider the application of the proposed TOPTW-PS, especially for the tourism sector (TTDP). Therefore, future research can apply the proposed algorithm in a practical application on an actual mobile device. In terms of the problem, multiple objectives can also help determine a considerably beneficial solution, such as setting an objective to maximize the total score and minimize the total cost. The potential applications of TOPTW-PS to other areas also include humanitarian logistic, personal medical service, and freelance workers. In addition, the option to randomize movement options among the four trajectories in S-DPSO can be explored. Finally, in order to find the best parameter values, automatic calibration tools, such as IRace (López-Ibáñez, Dubois-Lacoste, Cáceres, Birattari, & Stützle, 2016), can be considered.

Acknowledgements

This research was partially supported by the Ministry of Science and Technology of the Republic of China (Taiwan) under grant MOST 108-2221-E-011-051-MY3 and the Center for Cyber-physical System Innovation from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education of the Republic of China (Taiwan).