Jurnal Teknik Industri, Vol. 16, No. 1, Juni 2014, 9-16
ISSN 1411-2485 print / ISSN 2087-7439 online
DOI: 10.9744/jti.16.1.9-16

Applying Particle Swarm Optimization for Solving Team Orienteering Problem with Time Windows

The Jin Ai[1*], Evan Martinus Mahulae[1]

Abstract: The Team Orienteering Problem with Time Windows (TOPTW) is a transportation problem case that have a set of vertices with a score, service time, and the time windows, start and final at a depot location. A number of paths are constructed to maximize the total collected score by the vertices which is visited. Each vertice can be visited only once and the visit can only start during the time window of vertices. This paper proposes a Particle Swarm Optimization algorithm for solving the TOPTW, by defining a specific particle for representing the solution of TOPTW within the PSO algorithm and two alternatives, called PSO_TOPTW1 and PSO_TOPTW2, for translating the particle position to form the routes of the path. The performance of the proposed PSO algorithm is evaluated through some benchmark data problem available in the literature. The computational results show that the proposed PSO is able to produce sufficiently good TOPTW solutions that are comparable with corresponding solutions from other existing methods for solving the TOPTW.

Keywords: Particle swarm optimization, team orienteering problem, time windows, metaheuristics, solution methodology.

Introduction

For many organizations, including industry, transportation of vehicles to visit several locations for some specific purpose, i.e. delivery goods or service, can be considered as an important activity. During planning of this transportation activity, the organization needs to consider the visit time to each location, since each location to be visited usually has the earliest and the latest time of visits, or usually called the time windows. The vehicle can immediately stop in the location to accomplish its purpose, if and only if it arrives in the location during its time window. Otherwise, it has to wait in the location if it has arrived before the earliest time of visit of the location. In other case, the transportation vehicle cannot stop in the location if it arrives after the latest time of visit, that makes this transportation activity useless. Another practical aspect that needs consideration during the transportation planning is the condition in which case the organization should select locations to be visited among candidate locations due to time limitation. This decision problem can be solved using a simple computational procedure, i.e. enumeration method. However, this procedure consumes a lot of time when there are hundreds or thousands of locations to be evaluated.

This paper focuses on one case of transportation problem called the Team Orienteering Problem with Time Windows (TOPTW) by considering the conditions described above.

[1] Faculty of Industrial Technology, Industrial Engineering Department, Atma Jaya Yogyakarta University, Jl. Babarsari 44, Yogyakarta 55281. Indonesia, Email: jinai@mail.uajy.ac.id

*Corresponding author

This problem has root in another transportation problem so called the Orienteering Problem (OP). In the OP, there are a number of locations that usually called vertices, in which each of vertices has a definite score. The travel time is given between a pair of vertices. The decision of the OP is to determine a path starting from the vertex 1, visits to some vertices, and ending in the vertex N, in which the travel time of the path does not exceed its maximum travel time ($T_{\max}$). The objective function of the OP is to maximize the total collected score of the visited vertices (Golden *et al.* [1]; Vansteenwegen *et al.* [2]). A variant of OP is called the Orienteering Problem with Time Windows (OPTW), in which each vertice in this problem has a time window. So, a visit to each vertice can only be done within the available time windows (Kantor and Rosenwein [3]; Righini and Salani [4]). One application of OPTW is a travelling to some tourist destinations in one region. Each destination (vertex) has its advantages (score) and specific open and close time to visit (time windows). Since the tour has limited time (maximum travel time), the tour is not able to visit all destinations, therefore the tour guide needs to select which destinations to be visited and the route of the tour so that the tourist will obtain maximum advantages within the limited tour duration.

The TOPTW problem setting is similar to OPTW except that the TOPTW determines a set of paths instead of single path (Vansteenwegen *et al.* [5]). An example of TOPTW application is the scheduling of some medical representative staffs for visiting medical doctors in certain areas to offer new products, i.e. new medical devices. Usually they are assigned in a limited time ($T_{\max}$) and each doctor has

specific time to be visited (the doctor's time windows), so that their visiting schedule is very important to determine. Due to these limitations, they put a priority to the qualified and experienced doctors, i.e. doctor with a good 'score', so that the visiting sequence schedules of all staffs can cover the majority of good doctors in the area.

Although the mathematical model of the TOPTW can be formulated, i.e. in the paper of Montemanni and Gambardella [6], the problem is hard in nature in which the optimal solution can be found in reasonable computational time for small size problem only. Therefore, some researchers in the past already proposed some heuristics or applied some meta-heuristics for solving the TOPTW, such as iterated local search (Vansteenwegen *et al.* [5]), ant colony optimization (Montemanni and Gambardella [6]), hybridized evolutionary local search (Labadie *et al.* [7]), LP-based granular variable neighborhood search (Labadie *et al.* [8]), and simulated annealing (Lin and Yu [9]).

In the literature, it is found that particle swarm optimization (PSO), an emerging population based searching metaheuristics, has been successfully applied for solving some transportation problems such as vehicle routing problems and team orienteering problem. Several variants of vehicle routing problems that had been tackled with PSO are vehicle routing problem with time windows (Ai and Kachitvichyanukul [10]; Xu *et al.* [11]; Govindan *et al.* [12]), vehicle routing problem with simultaneous pickup and delivery (Ai and Kachitvichyanukul [13]; Goksal *et al.* [14]), and capacitated vehicle routing problem (Ai and Kachitvichyanukul, [15]; Kuo *et al.* [16]; Tlili *et al.* [17]). Two groups of author recently published their works on the application of PSO for solving the team orienteering problem (Dang *et al.* [18]; Ai *et al.* [19]). Since the TOPTW is very close with these problems, especially the vehicle routing problem with time windows (VRPTW) and the team orienteering problem (TOP), therefore, there is a high possibility to use PSO to solve the TOPTW. This paper proposes on how the PSO can be applied to solve the TOPTW and tests the proposed algorithm over some benchmark problems of TOPTW. To the authors knowledge, no other literature in the past applied PSO for solving the TOPTW.

The outline of this paper is as follows: In the first section the formal definition of TOPTW is presented. After that the PSO algorithm for TOPTW is proposed. The following section describes the computational experiments for evaluating the performance of the proposed algorithm. Finally, the conclusion of this study is presented with some suggestions for further research in this research area.

Methods

TOPTW Problem Formulation

As defined in some previous researches, i.e. Labadie *et al.* [8], the Team Orienteering Problem with Time Windows (TOPTW) can be formally defined as follow. Let us consider a set of visiting points $V = \{1,2,\dots,n\}$ plus a depot indexed by 0. Let $G = (V, A)$ be a directed graph where V is the set of vertices and A is the set of arcs. A positive integer score (profit) p_i is associated with each vertex, whereas $p_0 = 0$. Each vertex i has a time window $[e_i, l_i]$ where e_i is the earliest time and l_i is the latest time allowed for starting service at vertex i, whereas e_0 and l_0 are the earliest leaving time and the latest arrival time of each path to the depot, respectively. The maximum total travel time allowed to complete a tour is defined as $T_{\max}$. As consequences, it is assumed that $e_0 = 0$ and $l_0 = T_{\max}$. Let t_{ij} be the nonnegative travel time associated to each arc $(i,j) \in A$. It is assumed that the service time at vertex$i$ is already included in the t_{ij}. The problem looks for set of m paths with the following conditions:

(1) each path starts and ends at the depot,
(2) each vertex is visited at the most once by any path,
(3) the total duration of any path does not exceed $T_{\max}$,
(4) the visited vertices time windows are satisfied, and
(5) the total profit of visited vertices is maximized.

Proposed Particle Swarm Optimization for TOPTW

Particle Swarm Optimization (PSO) is a population-based stochastic optimization technique developed by Kennedy and Eberhart [20], inspired by the behavior of swarm organism such as bird flock, fish school, and bee swarm. PSO mimics the physical movement of individuals in the swarm to conduct the search mechanism of problem solution by two important parameters: position and velocity. A particle position, which is usually placed in multi-dimensional space, represents an alternative of problem solution. Velocity of particle is the driver of particle movement from one position to another. By moving to other position, another alternative of problem solution is evaluated. Therefore, the particle velocity expresses the searching capability of the problem solution. There are two important behaviors of the swarm organism that are formulated in the PSO, namely the cognitive behavior and the social behavior. The cognitive behavior is defined as the tendency of particle moving towards the best position ever visited by the particle, which is usually called personal best or pbest. While the social

behavior is defined as the tendency of particle moving towards the best position ever visited by all particles in the swarm, which is usually called personal best or pbest. The simplest version of particle movement can be stated as following equations:

$$\omega_{lh}(\tau+1) = w(\tau+1)\omega_{lh}(\tau) + c_p u[\psi_{lh} - \theta_{lh}(\tau)] + c_g u[\psi_{gh} - \theta_{lh}(\tau)] \quad (1)$$

$$\theta_{lh}(\tau+1) = \theta_{lh}(\tau) + \omega_{lh}(\tau+1) \quad (2)$$

where τ is iteration index, $l$is particle index, $h$is dimension index, $u$is uniform random number in interval [0,1], $w(\tau)$ is inertia weight in theτ^{th} iteration, $\omega_{lh}(\tau)$ is velocity of l^{th} particle at the h^{th} dimension in theτ^{th} iteration, $\theta_{lh}(\tau)$ is position of l^{th} particle at the h^{th} dimension in theτ^{th} iteration, $\psi_{lh}$is personal best position (pbest) of l^{th} particle at the h^{th} dimension, $\psi_{gh}$is global best position (gbest) at the h^{th} dimension,c_p is personal best acceleration constant, and c_g is global best acceleration constant. Equations 1 and 2 imply that the movement of particle in certain period of time is driven by three different directions that are: 1) follow its own way, 2) go towards its personal best position, and 3) go towards its global best position.

In general, the algorithm of PSO can be formally defined as follow:

1. initialization of particles, their position and initial velocity,
2. decode particles into problem solutions,
3. evaluate the quality of particles, based on their corresponding objective functions,
4. update pbest value,
5. update gbest value,
6. update velocity and position for each particle, i.e. based equations 1 and 2,
7. if the stopping criterion, i.e. maximum number of iteration, is reached, stop. Otherwise return to step 2.

Following this algorithm, the best problem solution is represented by the global best at the end of iteration. The details of PSO can be found in several textbooks, among others are Kennedy and Eberhart [20] and Clerc [21]. It is noted that this algorithm can be applied on various types of problems by defining how the particle represents the problem, which is usually called the solution representation, and how the particle can be translated into problem solution, which is usually called the decoding method. Therefore for applying PSO for TOPTW, we need to define the solution representation and the decoding method in the following subsections.

Solution Representation

Based on Ai *et al.* [19], the solution representation of TOPTW with n vertices is particle with n dimensions, in which each particle's dimension corresponds to each vertex, i.e. dimension 1 represents vertex 1, dimension 2 represents vertex 2, and so on. Particle position is assigned to be a real number and represents a priority of vertex on the decoding method. The smaller the position of particle, the higher the priority of the corresponding vertex. Later on the decoding steps, each vertex is evaluated to be inserted into the solution paths based on its priority.

Decoding Methods

The decoding method starts with the conversion of particle position into priority of vertex. Then, the vertex is evaluated for possibility to be inserted into the paths, one by one according to the priority. The solution of TOPTW is created when all the vertices already evaluated. We are proposing two alternatives of evaluation procedure. The first alternative is a simpler procedure, in which a vertex is evaluated to be inserted in the last sequence of each path, starting from the first path. If the insertion complies with the vertex's time window and path duration constraint, then the vertex is placed on the sequence. Otherwise, the vertex is evaluated to be inserted to the subsequent path. If the vertex cannot be inserted to any available paths, it implies that the vertex is decided not to be visited.

In the second alternative, a vertex is evaluated to be inserted into all possible sequences in all existing paths. Finally, the vertex is being inserted into a sequence in certain path that satisfies all time windows and total duration constraints and provides the smallest additional time. Figures 2 and 3 illustrate the first and second evaluation procedure, respectively. Later on PSO algorithm for TOPTW with the first and second alternative of evaluation procedures are called PSO_TOPTW1 and PSO_ TOPTW2, respectively.

dimension	1	2	3	4	5	6	7	8	9	10
position	0.52	2.69	1.03	0.15	1.94	3.17	1.29	3.67	0.76	2.38

sorted position	0.15	0.52	0.76	1.03	1.29	1.94	2.38	2.69	3.17	3.67
vertex no.	4	1	9	3	7	5	10	2	6	8
priority	1	2	3	4	5	6	7	8	9	10

Figure 1. Solution representation of TOPTW of 10 vertices and its conversion to priority of vertex

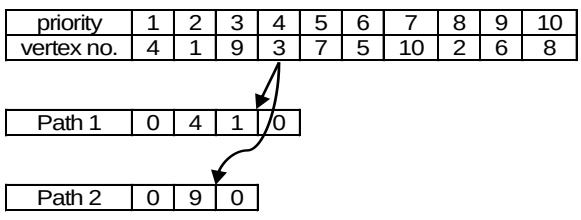

Figure 2. The first evaluation procedure

Table 1. Starting service time of the illustrated path 1

Vertex	0	4	1	3	0
Starting service time	0	15	20	37	45
Earliest service time	0	0	15	0	0
Latest service time	100	20	45	30	100

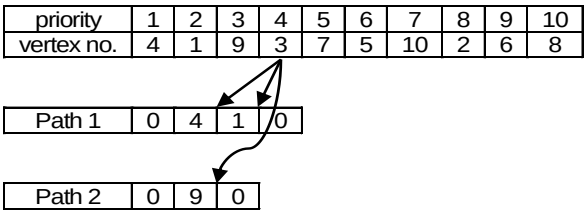

Figure 3. The second evaluation procedure

Table 2. Feasibility checking and additional time evaluation

Path	Feasibility	Additional Time
0 – 4 – 3 – 1 – 0	Yes	15
0 – 4 – 1 – 3 – 0	No	–
0 – 9 – 3 – 0	Yes	20

It is illustrated in Figure 2 for the first evaluation procedure that at this step vertex number 3 is evaluated to be inserted into the existing paths, in which currently the first path is 0 – 4 – 1 – 0 and the second path is 0 – 9 – 0. The evaluation starts from the first path, in which the vertex number 3 is to be inserted in the last sequence of this path, i.e. the path become 0 – 4 – 1 – 3 – 0. The travel time between vertex 1 and 3 is then considered to obtain the actual starting service time at vertex 3 and total travel time. Since the actual starting service time of the first path is presented at Table 1, the emerging path is not feasible as it violates the time windows constraint of vertex 3. Therefore, the first path remains the same and the evaluation of second path, i.e. 0 – 9 – 3 – 0 is needed. If this evaluation satisfies the time window and path duration constraint, then the path 2 is updated. Otherwise, vertex number 3 is decided not to be visited.

Figure 3 illustrates the second evaluation procedure for the same step as the first one. At first, three evaluations have to be evaluated, that are 0 – 4 – 3 – 1 – 0, 0 – 4 – 1 – 3 – 0, and 0 – 9 – 3 – 0, to identify all possible feasible paths. Then, among feasible paths the additional time due to adding vertex number 3 is evaluated that illustrated in Table 2. In this case, finally the first path is updated into 0 – 4 – 3 – 1 – 0, since this alternative is feasible and has the smallest additional time.

Initialization

In the initialization step of PSO algorithm, only one particle is initialized by using special method described below, while the others are initialized with random position. The initial position of special particle, i.e. the first particle, is obtained from the vertex score using following equation:

$$\theta_{1h}(0) = \theta_{\min} + (\theta_{\max} - \theta_{\min})/(p_h + 1) \quad (3)$$

where$\theta_{\min}, \theta_{\max}$ are the prescribed value of minimum and maximum of particle's position, and is the score of vertex h. By using this equation, vertex with higher score has the lower value of corresponding particle position value. As the consequence, the higher score vertex has a higher priority. The remaining particles are initialized with random positions by generating the position between the value of $\theta_{\min}$and $\theta_{\max}$.

Results and Discussions

Computational Experiments

Benchmark Data Instances

To conduct computational experiments, the existing TOPTW test instances data from previous researches, i.e. Righini and Salani [4], Montemanni and Gambardella [6], and Vansteenwegen *et al.* [2], are taken. A test instance describes a certain characteristic of TOPTW such as the number of vertices, the number of desired paths, maximum travel time ($T_{\max}$), and attributes of each vertex (coordinates, score, service time, time windows). These TOPTW instances are based on the Solomon [22] data instances of Vehicle Routing Problem with Time Windows, consisting of 100 vertices to be visited, and the Cordeau *et al.* [23] data instances of Multi Depot Vehicle Routing Problem, that ranges from 48 to 288 vertices to be visited. Righini and Salani [4] converted 48 Solomon's data instances into TOPTW instances called c10*, r10*, rc10* and 10 Cordeau's data instances into TOPTW instances called pr01–pr10. Montemanni and Gambardella [6] creates 37 TOPTW instances, 27 instances are converted from Solomon's data instances, which are called c20*, r20*, and rc20*, and the other instances are converted from Cordeau's instances, which are called pr11–pr20. Vansteenwegen *et al.* [2] also uses Solomon's and Cordeau's data instances and changes the number of paths in the problem. They also consider the optimal number of paths, in which the total collected score is the best among the other number of paths.

For comparison purpose, since the optimal solution for any TOPTW instances is not available, the result of the proposed method will be compared with the best known solution of each instances. In this paper, the best known solution is updated based on the result of Labadie [7] and Lin and Yu [9].

Experiment Setting

The proposed PSO algorithm for TOPTW, including the decoding methods and initialization procedure, are implemented using C# language and supported by PSO computational library called ET-Lib (Nguyen *et al.* [24]). It is noted that the ET-Lib uses a PSO variant called GLNPSO that has three different social behavior terms called global best, local best, and nearest neighbor best with its corresponding acceleration constant (c_g, c_l, and c_n). The main parameters of GLNPSO are taken from its user manual (Nguyen *et al.* [24]), that are$c_p = 1$, $c_g = 1$, $c_l = 1$, and $c_n = 1$. All the test instances are run on a computer with an Intel Pentium dual core 2.70 GHz CPU and 2 GB RAM. For each instance, 10 replications of the PSO algorithm runs are conducted.

Parameters Optimization

It is well known in the PSO literature that its parameters have an influence in the solution quality, so that these parameters need to be optimized. In this experiment, an optimization process is conducted to select the number of particles and the number of iterations in such a way that the algorithm is able to provide minimum average deviation from the best known solution at minimum computational time. Test instances c101 with 4 paths is used for the optimization process to evaluate the combination of 30, 50, 100, and 200 particles with 500, 1000, and 2000 iterations. The average deviation is calculated based on the deviation of each iterations, in which the deviation is defined as the difference between the total score of best known solution and the total score of PSO solution.

Empirically, the larger number of particles and number of iterations the better solution can be obtained by the algorithm. However, it is followed by the longer computational times. This experiment also confirms this statement, in which the best result is obtained from the combination of 200 particles and 2000 iterations with the longest computational time. For computational efficiency, we will determine combination of parameters that can provide result that is statistically similar with the result of combining 200 particles and 2000 iterations but with smallest computational time. It is found that combination of 100 particles and 1000 iterations is the preferred one. In addition, the result from combination of 30 particles and 1000 iterations only differs by 0.6% from the best result but significantly uses smaller computational time. Therefore, considering the efficiency of the computational time, both combinations (30 particles and 1000 iterations, 100 particles and 1000 iterations) are used for all test instances and both evaluation procedures inside the decoding method.

Computational Results

The PSO_TOPTW1 and PSO_TOPTW2 results are compared to the results from existing algorithms for TOPTW, which are Simulated Annealing (SSA and FSA) of Lin and Yu [9], Iterated Local Search (ILS)of Vansteenwegen *et al.* [2], LP-based GVNS (VNS) of Labadie *et al.* [8], Ant Colony System (ACO) of Montemanni and Gambardella [6], and GRASP-ELS of Labadie *et al.* [7]. In particular the results of our proposed algorithm are categorized into 3 parts, namely PSO_TOPTW1_30_1000, PSO_TOPTW2 _30_1000, and PSO_TOPT W2_100_ 1000. The two numbers after the algorithm name indicate number of particles and number of iterations, respectively.

Table 3, 4, and 5 shows the results of the TOPTW1_ 30_1000, TOPTW2_30_1000, and TOPTW2_100_ 1000. Results are grouped into the each instance data set c10*, r10*, c20*, r20*, rc20*, pr01–10, pr11–20. Columns I and II shows the number of paths and data sets used. Column III–V summarize the solution of respective group, in terms of the average (Avg.), standard deviation (S.D.), and minimum (Min.) percentage deviation of 10 PSO replications. The percentage deviation (*dev*) is calculated using formula 4, where f_{PSO} is the total profit of PSO solution and f_{BKS} is the total profit of the best known solution from existing algorithms for TOPTW. Column VI–VII show the average (Avg.) and standard deviation (S.D.) of computational time.

$$dev = \frac{f_{BKS} - f_{PSO}}{f_{BKS}} \times 100\% \quad (4)$$

It is shown from Table 3 and 4 that the average percentage deviation of PSO_TOPTW1_30_1000 is bigger than PSO_TOPTW2_30_1000, while the computational time of PSO_TOPTW1_30_1000 is smaller than PSO_TOPTW2_30_1000. These results imply that the decoding process in PSO_TOPTW2, which is more complex than PSO_TOPTW1, can produce better solutions. Table 5 shows that increasing number of particles is able to yield smaller deviation but requires more computational effort, since the results from PSO_TOPTW2_100_1000 is better than the results from PSO_TOPTW2_ 30_1000 and the computational time of PSO_TOPT W2_100_1000 is bigger than the computational time of PSO_TOPTW2_30_1000.

In addition to the result presented in Tables 3–5, the TOPTW1_30_1000 is able to produce 11 solution of instances, among 304 instances used in the computational tests, reaching its corresponding best known

solution. The TOPTW2_30_1000 is able to produce 74 solution of instances reaching its corresponding best known solution. The TOPTW2_100_1000 is able to produce 85 solution of instances reaching its corresponding best known solution. Combined all approaches together, the PSO algorithms in this research are able to produce 88 solution of instances that are similar with its corresponding best known solutions. Furthermore, the TOPTW2_100_1000 is able to provide 1 solution of instance that is outperform its corresponding best known solution.

Table 3. Summarize of PSO_TOPTW1_30_1000 results

No. of Paths	Inst Group	%Dev			CPU Time	
		Avg.	S.D.	Min	Avg.	S.D.
1	c10*	4.04%	2.13%	1.70%	10.81	0.21
	c20*	15.01%	3.01%	10.37%	13.58	0.28
	r10*	11.64%	4.52%	5.99%	11.44	0.27
	r20*	26.99%	2.99%	22.85%	14.28	0.46
	rc10*	14.32%	5.69%	6.04%	12.41	1.46
	rc20*	31.24%	4.38%	25.24%	13.43	0.85
	pr01-10	31.72%	5.12%	23.53%	91.60	9.91
	pr11-20	31.88%	4.44%	25.10%	87.99	9.42
2	c10*	6.91%	2.21%	3.73%	13.74	0.35
	c20*	17.49%	2.36%	13.98%	16.78	0.41
	r10*	18.20%	3.77%	12.36%	13.82	0.44
	r20*	23.97%	2.43%	20.18%	17.59	0.39
	rc10*	16.35%	3.88%	10.33%	16.68	3.08
	rc20*	31.57%	2.57%	28.09%	16.90	0.45
	pr01-10	32.81%	3.93%	27.36%	23.06	0.50
	pr11-20	36.63%	3.19%	32.13%	21.70	0.60
3	c10*	9.13%	2.10%	5.79%	10.00	0.06
	c20*	17.08%	2.06%	14.16%	11.13	0.08
	r10*	21.73%	3.23%	16.40%	9.88	0.26
	r20*	11.46%	2.01%	8.54%	22.89	35.96
	rc10*	18.17%	4.26%	11.50%	9.51	0.51
	rc20*	22.03%	2.36%	18.29%	11.11	0.17
	pr01-10	32.85%	3.38%	27.84%	27.68	0.43
	pr11-20	37.02%	2.37%	33.35%	26.71	0.35
4	c10*	9.98%	1.61%	7.64%	13.12	0.28
	c20*	7.24%	1.84%	4.63%	12.84	0.28
	r10*	22.67%	3.07%	17.68%	22.01	32.96
	r20*	3.51%	1.11%	1.93%	13.02	0.20
	rc10*	21.68%	3.45%	16.00%	11.30	0.48
	rc20*	11.26%	1.88%	8.77%	12.90	0.18
	pr01-10	31.24%	2.94%	26.29%	41.62	1.02
	pr11-20	34.55%	2.41%	30.75%	37.96	0.50
optimal	c10*	7.45%	0.90%	6.02%	28.88	0.29
	c20*	7.24%	1.84%	4.63%	12.84	0.28
	r10*	13.09%	1.62%	10.55%	32.08	0.45
	r20*	16.21%	2.19%	13.19%	25.38	36.11
	rc10*	16.40%	1.88%	13.22%	31.37	0.37
	rc20*	12.79%	1.87%	10.03%	12.25	0.30
	pr01-10	17.14%	1.69%	14.68%	191.12	2.38

Table 4. Summarize of PSO_TOPTW2_30_1000 results

No. of Paths	Inst Group	%Dev			CPU Time	
		Avg.	S.D.	Min	Avg.	S.D.
1	c10*	0.73%	0.44%	0.26%	25.01	0.58
	c20*	2.13%	0.92%	0.92%	101.70	2.48
	r10*	4.04%	2.24%	0.98%	23.02	1.46
	r20*	7.88%	1.75%	5.21%	154.24	8.69
	rc10*	3.92%	2.96%	0.62%	19.48	1.11
	rc20*	9.45%	2.57%	5.79%	118.24	8.40
	pr01-10	13.99%	3.30%	9.00%	91.60	9.91
	pr11-20	19.91%	3.64%	14.45%	87.99	9.42
2	c10*	1.50%	1.06%	0.30%	33.90	1.01
	c20*	3.85%	0.78%	2.62%	137.30	37.45
	r10*	8.01%	2.92%	3.09%	28.85	1.71
	r20*	6.31%	1.31%	4.17%	154.66	6.08
	rc10*	6.88%	2.65%	3.35%	26.14	1.21
	rc20*	11.47%	2.44%	8.36%	125.89	7.70
	pr01-10	18.09%	3.16%	12.50%	119.67	9.50
	pr11-20	23.87%	3.26%	18.18%	103.41	9.28
3	c10*	2.86%	1.07%	1.37%	75.73	97.39
	c20*	4.63%	0.83%	3.31%	159.09	38.36
	r10*	9.86%	2.09%	6.40%	36.88	1.58
	r20*	0.68%	0.22%	0.43%	149.51	3.42
	rc10*	8.50%	3.04%	4.59%	33.35	1.54
	rc20*	4.04%	0.96%	2.40%	130.58	4.98
	pr01-10	19.44%	2.83%	15.08%	142.87	9.53
	pr11-20	23.32%	3.05%	18.55%	135.62	10.68
4	c10*	3.86%	0.93%	2.49%	52.69	1.05
	c20*	0.00%	0.00%	0.00%	229.38	0.73
	r10*	11.38%	2.41%	7.61%	221.68	463.79
	r20*	0.01%	0.01%	0.00%	376.64	396.65
	rc10*	10.85%	2.59%	7.36%	70.00	3.22
	rc20*	0.21%	0.13%	0.00%	220.14	3.84
	pr01-10	18.74%	2.81%	14.76%	169.99	12.37
	pr11-20	21.45%	2.33%	16.97%	161.13	8.64
optimal	c10*	3.27%	0.56%	2.27%	149.93	16.95
	c20*	0.00%	0.00%	0.00%	229.38	0.73
	r10*	5.76%	0.89%	4.38%	98.25	1.93
	r20*	2.11%	0.47%	1.45%	278.42	373.95
	rc10*	7.17%	1.40%	5.15%	174.11	11.01
	rc20*	1.18%	0.36%	0.57%	208.55	3.70
	pr01-10	7.40%	1.05%	5.65%	456.72	3.83

Conclusion

This paper tries to solve the Team Orienteering Problem With Time Windows (TOPTW) by using Particle Swarm Optimization (PSO) algorithm. A specific particle is defined for representing the solution of TOPTW within the PSO algorithm, in which the particle can be translated into priority of vertices and later on the priority of vertices can be utilized to form the routes of the path. In this paper, there are two alternatives of method to form routes, which are called PSO_TOPTW1 and PSO_TOPTW2. PSO_TOPTW1 test the priority of each vertices in the last sequence of the route.

Table 5. Summarize of PSO_TOPTW2_100_1000 results

No. of Paths	Inst Group	%Dev Avg.	%Dev S.D.	%Dev Min	CPU Time Avg.	CPU Time S.D.
1	c10*	0.59%	0.22%	0.26%	109.40	1.91
	c20*	1.62%	0.62%	0.92%	363.23	8.09
	r10*	2.46%	1.64%	0.75%	116.27	4.57
	r20*	5.39%	1.63%	2.71%	665.94	32.22
	rc10*	1.91%	1.76%	0.46%	155.64	11.51
	rc20*	7.25%	2.23%	4.16%	601.41	42.47
	pr01-10	13.09%	2.84%	8.60%	437.84	44.30
	pr11-20	15.50%	3.39%	9.96%	614.59	65.87
2	c10*	0.90%	0.65%	0.15%	263.52	5.78
	c20*	3.32%	0.58%	2.37%	1125.31	2078.69
	r10*	5.19%	1.67%	2.79%	130.25	63.64
	r20*	4.79%	1.06%	3.17%	620.07	242.10
	rc10*	4.18%	2.03%	1.90%	272.56	542.43
	rc20*	8.80%	1.64%	6.32%	532.69	248.19
	pr01-10	14.56%	3.10%	9.34%	460.87	119.62
	pr11-20	19.75%	3.00%	15.15%	432.63	34.89
3	c10*	2.20%	0.87%	1.15%	334.40	5.95
	c20*	4.19%	0.76%	3.18%	1184.06	1989.05
	r10*	7.70%	1.99%	4.98%	168.31	90.05
	r20*	0.41%	0.17%	0.24%	929.30	1280.01
	rc10*	6.37%	2.32%	3.50%	304.36	552.52
	rc20*	2.96%	0.91%	1.62%	551.56	251.51
	pr01-10	18.93%	6.44%	12.03%	676.26	161.03
	pr11-20	20.40%	2.96%	15.97%	647.08	46.35
4	c10*	3.07%	0.89%	1.61%	431.75	8.85
	c20*	0.00%	0.00%	0.00%	1278.76	2417.69
	r10*	9.42%	2.10%	6.46%	243.30	249.63
	r20*	0.00%	0.00%	0.00%	525.16	2.97
	rc10*	8.33%	2.43%	5.09%	336.43	511.76
	rc20*	0.10%	0.09%	0.00%	504.88	174.41
	pr01-10	16.28%	2.19%	12.60%	1122.58	499.17
	pr11-20	19.14%	2.11%	16.21%	1709.08	2251.29
optimal	c10*	2.80%	0.52%	2.09%	378.33	5.91
	c20*	0.00%	0.00%	0.00%	1278.76	2417.69
	r10*	4.63%	0.78%	3.60%	469.01	23.47
	r20*	1.73%	0.38%	1.26%	344.49	12.78
	rc10*	6.52%	1.45%	4.76%	338.90	5.99
	rc20*	1.08%	0.34%	0.57%	363.56	4.74
	pr01-10	6.44%	1.20%	4.52%	1818.35	143.44

Vertices will be placed on route if satisfy the time window and duration less than T_max, conversely, vertices tested on the next path. PSO_TOPTW2 is more complex. The priority of vertices is tested in every sequences of the route, starting from the last sequence. Vertices will be placed on the route if they satisfy time window and have the best time duration, conversely, vertices tested on the next path.

The result of program is divided into three parts, TOPTW1_30_1000, TOPTW2_30_1000, TOPTW2_100_1000, which distinguishes the alternative decoding method, the number of particles, and the number of iterations that used. TOPTW1_30_1000 has a biggest average percentage deviation smallest computational time, otherwise, TOPTW2_100_1000 has a smallest average percentage deviation and the biggest computational time. Overall, this research can produce 88 existing best known solutions and improve 1 best known solution. Comparison with other optimization method shows that the proposed method is good enough.

The authors believe that it is still possible to improve the performance of the proposed method. However, some further research is required, such as in the area of parameter optimization, decoding alternatives, hybridization of method, and computer programming implementation. Selection of PSO parameter may improve the performance of the algorithm. There are also room for improving the decoding methods. A modification in algorithm, like multi-swarm methods, or a combination with other optimization algorithm, like local search, can be conducted to get better result. The improvement should address also the issue of shortening the computational times of the proposed method via good computer programming implementation. Future research direction also open for applying the PSO for other orienteering problem variants, such as capacitated team orienteering problem.

Acknowledgment

The authors express thanks to High Performance Computing Group at Asian Institute of Technology, Thailand, for providing the Object Library for Evolutionary Techniques (ET-Lib) version 1.0 used in this research. The authors also thank two anonymous reviewers that providing useful comments to help improving the presentation of this paper.

References

1. Golden, B. L., Levy, L., and Vohra, R., The Orienteering Problem, *Naval Research Logistics*, 34(3), 1987, pp. 307-318.
2. Vansteenwegen, P., Souffriau, W., Berghe, G. V., and van Oudheusden, D. V., The Orienteering Problem: A Survey, *European Journal of Operation Research*, 209(1), 2011, pp. 1-10.
3. Kantor, M. G., and Rosenwein, M. B., The Orienteering Problem with Time Windows, *Journal of the Operational Research Society*, 43(6), 1992, pp. 629-635.
4. Righini, G., and Salani, M., Decremental State Space Relaxation Strategies and Initialization Heuristics for Solving the Orienteering Problem with Time Windows with Dynamic Programming, *Computers & Operations Research*, 36(4), 2009, pp. 1191-1203.
5. Vansteenwegen, P., Souffriau, W., van den Berghe, G., and van Oudheusden, D., Iterated Local Search for the Team Orienteering Problem with Time Windows, *Computers & Operations Research*, 36(12), 2009, pp. 3281-3290.

6. Montemanni, R., and Gambardella, L. M., An Ant Colony System for Team Orienteering Problems with Time Windows, *Foundations of Computing and Decision Sciences*, 34, 2009, pp. 287-306.
7. Labadie, N., Melechovský, J., and Calvo, R. W., Hybridized Evolutionary Local Search Algorithm for the Team Orienteering Problem with Time Windows, *Journal of Heuristics*, 17(6), 2011, pp. 729-753.
8. Labadie, N., Mansini, R., Melechovský, J., and Wolfler Calvo, R., The Team Orienteering Problem with Time Windows: An lp-Based Granular Variable Neighborhood Search, *European Journal of Operational Research*, 220(1), 2012, pp. 15-27.
9. Lin, S. W., and Yu, V. F., A Simulated Annealing Heuristic for the Team Orienteering Problem with Time Windows, *European Journal of Operational Research*, 217(1), 2012, pp. 94-107.
10. Ai, T. J., and Kachitvichyanukul, V., A Particle Swarm Optimisation for Vehicle Routing Problem with Time Windows, *International Journal of Operational Research*, 6(4), 2009, pp. 519-537.
11. Xu, J., Yan, F., and Li, S., Vehicle Routing Optimization with Soft Time Windows in a Fuzzy Random Environment, *Transportation Research Part E: Logistics and Transportation Review*, 47(6), 2011, pp. 1075-1091.
12. Govindan, K. , Jafarian, A. , Khodaverdi, R., and Devika, K., Two-Echelon Multiple-Vehicle Location–Routing Problem with Time Windows for Optimization of Sustainable Supply Chain Network of Perishable Food, *International Journal of Production Economics*, 152, 2013,pp.9-28.
13. Ai, T. J., and Kachitvichyanukul, V., A Particle Swarm Optimization for the Vehicle Routing Problem with Simultaneous Pickup and Delivery, *Computers & Operations Research*, 36(5), 2009, pp. 1693-1702.
14. Goksal, F. P., Karaoglan, I., and Altiparmak, F., A Hybrid Discrete Particle Swarm Optimization for Vehicle Routing Problem with Simultaneous Pickup and Delivery, *Computers & Industrial Engineering*, 65(1), 2013, pp. 39-53.
15. Ai, T. J., and Kachitvichyanukul, V., Particle Swarm Optimization and Two Solution Representations for Solving the Capacitated Vehicle Routing Problem, *Computers & Industrial Engineering*, 56(1), 2009, pp. 380-387.
16. Kuo, R. J., Zulvia, F. E., and Suryadi, K., Hybrid Particle Swarm Optimization with Genetic Algorithm for Solving Capacitated Vehicle Routing Problem with Fuzzy Demand – A Case Study on Garbage Collection System. *Applied Mathematics and Computation*, 219(5), 2012, pp. 2574-2588.
17. Tlili, T., Faiz, S., and Krichen, S., A Hybrid Metaheuristic for the Distance-Constrained Capacitated Vehicle Routing Problem, *Procedia - Social and Behavioral Sciences*, 109, 2014, pp. 779-783.
18. Dang, D. C., Guibadj, R. N., and Moukrim, A, An Effective PSO-Inspired Algorithm for the Team Orienteering Problem, *European Journal of Operational Research*, 229(2), 2013, pp. 332-344.
19. Ai, T. J., Pribadi, J. S., and Ariyono, V., Solving the Team Orienteering Problem with Particle Swarm Optimization, *Industrial Engineering & Management Systems*, 12(3), September 2013, pp.198-206.
20. Kennedy, J., and Eberhart, R. C., *Swarm Intelligence*, San Francisco: Morgan Kaufmann Publishers, 2001.
21. Clerc, M., *Particle Swarm Optimization*, London: ISTE, 2006.
22. Solomon, M.M., Algorithms for the Vehicle Routing and Scheduling Problems with Window Constraints, *Operations Research*, 35(2), 1987, pp. 254-265.
23. Cordeau, J. F., Gendreau, M., and Laporte, G., A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems, *Networks*, 30(2), 1997, pp. 105-119.
24. Nguyen, S., Ai, T. J., and Kachitvichyanukul, V., Object Library for Evolutionary Techniques ETLib: User's Guide, *High Performance Computing Group, Asian Institute of Technology, Thailand*, 2010.