

Solving the Team Orienteering Problem with Particle Swarm Optimization

The Jin Ai*, Jeffry Setyawan Pribadi, Vincensius Ariyono

Department of Industrial Engineering, Faculty of Industrial Technology, Universitas Atma Jaya Yogyakarta, Indonesia

(Received: January 17, 2013 / Revised: August 29, 2013 / Accepted: September 10, 2013)

ABSTRACT

The team orienteering problem (TOP) or the multiple tour maximum collection problem can be considered as a generic model that can be applied to a number of challenging applications in logistics, tourism, and other fields. This problem is generally defined as the problem of determining P paths, in which the traveling time of each path is limited by T_{max} that maximizes the total collected score. In the TOP, a set of N vertices i is given, each with a score S_i . The starting point (vertex 1) and the end point (vertex N) of all paths are fixed. The time t_{ij} needed to travel from vertex i to j is known for all vertices. Some exact and heuristics approaches had been proposed in the past for solving the TOP. This paper proposes a new solution methodology for solving the TOP using the particle swarm optimization, especially by proposing a solution representation and its decoding method. The performance of the proposed algorithm is then evaluated using several benchmark datasets for the TOP. The computational results show that the proposed algorithm using specific settings is capable of finding good solution for the corresponding TOP instance.

Keywords: Team Orienteering Problem, Particle Swarm Optimization, Solution Representation, Algorithm, Computational Method, Metaheuristics

* Corresponding Author, E-mail: jinaai@mail.uajy.ac.id

1. INTRODUCTION

The orienteering problem (OP) is a generic problem for determining a single path in order to maximize the total collected score by visiting some vertices through the path, in which a set of vertices is given, each with score; the starting and ending point of the path is given, the travel time from any vertex to any vertex (t_{ij}) is known, and the traveling time of the path is limited by T_{max} . Tsiligirides (1984) mentioned that OP can be applied to a special case of the traveling salesman problem (TSP) in which the salesman does not have enough time to visit all possible cities. He knows the number of expected sales in each city and wants to maximize total sales, while keeping the total travel time limited to a day or a week. Soufriaoui *et al.* (2008) applied the OP for solving a tourist route problem. Since it is often impossible to visit all the tourist attractions in a region visited by the tourist, they

have to select what they believe to be the most valuable attraction in the region.

The team orienteering problem (TOP) is an OP with goal to determine several paths, each is limited by T_{max} that maximizes the total collected score. Butt and Cavalier (1994) described a TOP application of athlete recruitment from high schools. A recruiter has to visit several schools, limited by number of days. He can assign a score for each school, based on its recruiting potential. Then, the TOP can help the recruiter to choose the schools to visit each day to maximize the total recruiting potential. The total number of paths in the TOP is corresponding to the total number of recruiting days, and each path in the TOP is corresponding to the recruiter daily route visiting the selected schools.

Since the TOP is an NP-hard problem, some exact and heuristics approaches had been proposed in the past for solving the TOP. However, the capability of particle

swarm optimization (PSO) has not been explored much in order to solve the TOP. Therefore, this paper tries to solve the TOP by using a PSO algorithm. The structure of this paper is as follows: Section 2 provides literature review on TOP. Section 3 describes the PSO for solving the TOP. Section 4 presents the computational result, and the last section concludes the paper and gives suggestions for further research.

2. LITERATURE REVIEW

Some researchers in the past had proposed methodology for solving TOP as reviewed by Vansteenwegen *et al.* (2011) who summarized existing and up-to-date research on TOP. They listed several algorithms including heuristics, Tabu search, variable neighborhood search, ant colony optimization, and greedy randomized adaptive search procedure (GRASP) with path relinking that has been successfully applied to TOP.

Chao *et al.* (1996a) presented the first heuristic algorithm for solving the TOP which is similar to their five-step heuristic algorithm for OP (Chao *et al.*, 1996b). They also contributed in collecting datasets of OP and TOP cases that are being used for comparing the performance of OP and TOP algorithms. It is noted that they organized the datasets into 7 sets.

Tang and Miller-Hooks (2005) proposed a Tabu search heuristic (TMH) algorithm for solving the TOP. They compared the Tabu search method with the method of Chao *et al.* (1996a) and found that the Tabu search was able to produce better solution than the Chao *et al.* (1996a) method. Archetti *et al.* (2007) also proposed some variants of Tabu search and variable neighborhood search for solving the TOP, which are called Tabu search with penalty strategy (GTP), Tabu search with feasible strategy (GTF), fast variable neighborhood search (FVF), and slow variable neighborhood search (SVF).

Another approach for solving TOP is proposed by Ke *et al.* (2008) by using ant colony optimization. Based on the method to construct feasible solutions, they proposed four variations of ant colony algorithm, which are sequential (ASe), deterministic-concurrent (ADC), random-concurrent (ARC), and simultaneous ant colony algorithms (ASi).

Two contributions from Vansteenwegen *et al.* (2009a, 2009b) are able to solve the TOP within seconds of computational time. For solving the TOP, Vansteenwegen *et al.* (2009a) implemented the guided local search (GLS) and Vansteenwegen *et al.* (2009b) implemented skewed variable neighborhood search (SVNS).

Souffriau *et al.* (2010) proposed two GRASP with path relinking algorithms for solving TOP, which are fast path relinking (FPR) and slow path relinking (SPR). The difference between these two algorithms is only the stop-

Table 1. Summary of results after Souffriau *et al.* (2010)

Algo-rithm	# Best	Avg. gap (%)	Avg. CPU (s)	Reference
TMH	34	1.47	336.6	Tang and Miller-Hooks (2005)
GTP	69	0.54	100.0	Archetti <i>et al.</i> (2007)
GTF	94	0.25	146.6	Archetti <i>et al.</i> (2007)
FVF	97	0.22	18.9	Archetti <i>et al.</i> (2007)
SVF	128	0.06	268.6	Archetti <i>et al.</i> (2007)
ASe	130	0.11	252.3	Ke <i>et al.</i> (2008)
ADC	80	0.42	213.8	Ke <i>et al.</i> (2008)
ARC	81	0.47	204.8	Ke <i>et al.</i> (2008)
ASi	84	0.39	215.0	Ke <i>et al.</i> (2008)
GLS	21	2.71	7.8	Vansteenwegen <i>et al.</i> (2009a)
SVNS	44	1.08	3.8	Vansteenwegen <i>et al.</i> (2009b)
FPR	78	0.46	5.0	Souffriau <i>et al.</i> (2010)
SPR	131	0.06	212.4	Souffriau <i>et al.</i> (2010)

TMH: Tabu search heuristic, GTP: Tabu search with penalty strategy, GTF: Tabu search with feasible strategy, FVF: fast variable neighborhood search, SVF: slow variable neighborhood search, ASe: ant colony sequential, ADC: ant colony deterministic-concurrent, ARC: ant colony random-concurrent, ASi: ant colony simultaneous, GLS: guided local search, SVNS: skewed variable neighborhood search, FPR: fast path relinking, SPR: slow path relinking.

ping criterion. The SPR requires more iterations than FPR. They also compared their algorithms with above mentioned methods for solving TOP including TMH, GTP, GTF, FVF, SVF, ASe, ADC, ARC, ASi, GLS, and SVNS using TOP benchmark datasets of Chao *et al.* (1996b) number 4 to 7 consist of 157 cases. Their comparison is reproduced here in Table 1. From Table 1, they claimed that that SPR is the best method for solving TOP, since it can produce 131 best known solution with average gap 0.06% from the best known results, consuming in average 212.4 seconds of computational time.

Recently, Muthuswamy and Lam (2011) proposed a discrete particle swarm optimization (DPSO) for solving TOP. They used discrete version of PSO, in which the particle position is represented in the domain of integer number, combined with reduced variable neighborhood search and 2-opt as the local search method.

This paper proposes a PSO algorithm for solving TOP, based on the simplest version of continuous PSO, in which the particle position is represented as real number and implemented purely from any local search. Therefore, the particle movement mechanism of this PSO that will be described in Section 3 is totally different with DPSO of Muthuswamy and Lam (2011). In order to do benchmarking, the result of Souffriau *et al.* (2010) will be referenced so that the result of proposed PSO can be placed in the performance of existing approaches completely, in which the similar TOP benchmark datasets

number 4 to 7 of Chao *et al.* (1996b) will be used as the case problems. In addition, this paper also provides a comparison between the proposed continuous PSO with the existing DPSO for solving the TOP.

3. PSO FOR SOLVING TOP

3.1 Particle Swarm Optimization

PSO is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995, inspired by social behavior of swarm organism such as flock of bird or school of fish (Hu, 2006).

Clerc (2004) stated that the basic principle of PSO is a set of moving particles placed inside the search space. Each particle has the following characteristics:

- 1) It has a position (θ) and velocity (ω). Its positions are multi-dimensional, and can be translated or decoded into problem specific solution, which will evaluate its corresponding objective function for minimization or maximization. The particle is moving from a position into different position following its velocity.
- 2) It knows its position and the objective function value of this position, and it also remembers its best previous position or usually called as personal best or pbest (ψ).
- 3) It knows the best of all particles' personal best or usually called as global best or gbest.

The movement of particle in certain period of time or iteration 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. Therefore the particle movement can be formulated 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 the τ^{th} iteration, $\omega_{lh}(\tau)$ is velocity of l^{th} particle at the h^{th} dimension in the τ^{th} iteration, $\theta_{lh}(\tau)$ is position of l^{th} particle at the h^{th} dimension in the τ^{th} iteration, ψ_{lh} is personal best position (pbest) of l^{th} particle at the h^{th} dimension, ψ_{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.

Eq. (2) shows that the particle position of next period is obtained from the sum of the current position with the velocity of the next period. Eq.(1) shows that the velocity

of next period obtained from the sum of the multiplication of inertia, cognitive, or social weights (w , c_p , and c_g) with current velocity, pbest, and gbest. Generally, the algorithm of PSO is defined as follow:

- Initialization: Determine the number of particles (N), the particle's position (can be random or defined), and particle's velocity.
- Decode particles into solution.
- Evaluate the particles, based on the objective function.
- Update pbest value,
- Update gbest value,
- Update velocity and position for each particle,
- If the stopping criterion is reached, stop. Otherwise return to decoding step.

The PSO algorithm has been applied for solving various optimization problems, such as TSP (Clerc, 2004), protein motif discovery (Chang *et al.*, 2004), as well as the vehicle routing with simultaneous pickup and delivery by Ai and Kachitvichyanukul (2009).

3.2 Solution Representation

In a simple functional optimization, the dimension of particles in the PSO is represented by the number of variables used in the function, and can directly be decoded as a value of the variables. However, in this research, particle's position cannot directly be decoded or represents the solution, because the solution of TOP is not just a value of variables, but is a sequence of vertices. Therefore, it needs a specific step to decode a particle position into a TOP solution that can be explained as follow:

- 1) The particle's dimension represents number of vertices in TOP, dimension 1 represent vertex 1, and so on.
- 2) The particle's position is a real number and represents priority of the vertex on the decoding step. The smaller the position will result in a higher priority on the vertex.
- 3) Based on the priority, each vertex is evaluated to be inserted to the solution path.

The first and last vertex are excluded from the priority

Particle										
h^{th} dimension	0	1	2	3	4	5	6	7	8	9
Position (θ_{lh})	7.4	15	3.1	5	17	24	30	8.3	4.7	14
Value (v_h)	7.4	15	3.1	5	17	24	30	8.3	4.7	14

Vertex										
Priority (v_{ph})	0	2	8	3	7	1	4	5	6	9

Path 1 (s_{1h})	0	2	8	9
Path 2 (s_{2h})	0	3	7	9

Figure 1. Illustration of decoding process.

setting of the vertex, since they are representing the starting and ending point of each path that cannot be changed. Figure 1 shows an illustration of the decoding process of the PSO for solving TOP with 10 vertices. In Figure 1, the sequence of the paths is simply constructed from the vertex priority. In this paper, before a vertex is permanently assigned to a path, it has to be evaluated based on the total service time of the path. If the time needed exceeds the limit (T_{max}), the vertex cannot be inserted into that path.

3.3 Initialization

In the initialization stage, the initial position of the first particle is obtained from score of each vertex. The score from each vertex is copied and converted into variable $SPar$, by using equation:

$$SPar_j = \min + \text{range} / (S_j + 1) \quad (3)$$

where $\text{range} = \text{max} - \text{min}$, \min is the minimum x or y coordinate for all vertices, max is the maximum x or y coordinate for all vertices, and S_j = score for vertex j . The score is added by 1 to avoid division by zero. By using Eq. (3), vertex with higher score will be assigned in lower value of $SPar$. Those process will make higher score vertex has higher priority.

The remaining particles are initialized with random positions in which the position is generated between the value of \min and max . Figure 2 illustrates the initialization of the first particle.

3.4 Decoding Method

The decoding method is divided into 2 steps, which are: construction of vertex's priority and evaluation and assignment of vertex.

Construction of vertex's priority is very simple. The value of dimension in particle's position is sorted in ascending order, with the smallest position will be the first priority (starting and ending point is excluded). The algorithm of the construction stage is:

Min	2
Max	35
Range	33
Vertex	0 1 2 3 4 5 6 7 8 9
Score	0 32 50 15 32 1 72 36 98 0
SPar	35 3 2.6 4.1 3 19 2.5 2.9 2.3 35
Particle 1	
dimen	0 1 2 3 4 5 6 7 8 9
value	35 3 2.6 4.1 3 19 2.5 2.9 2.3 35
Priori	0 8 6 2 7 1 4 3 5 9

Figure 2. Illustration of initialization.

1. Copy the value of particle's dimension to variable v , and dimension's index to v_p .
 $v_h = \theta_{lh}, v_{ph} = h; h = 0, \dots, N-1$.
2. Compare the value of each dimension with another dimension.
 For $i = 1$ to $N-2$, compare v_i with $v_j; j = i + 1, \dots, N-2$
3. If $v_i > v_j$, swap the value of v_i and v_j, v_{pi} and v_{pj} .
4. Repeat step 2 until $i = N-2$.

where N = number of vertices, v_{ph} = h^{th} priority of vertex, and θ_{lh} = value of h^{th} dimension in particle $l; h = 0, \dots, N-1$. Within the construction algorithm, v_p represent the priority of vertices.

The evaluation of vertex is a step to evaluate each vertex, one by one based on priority, to be assigned into a specific sequence in a path. The possibility of each vertex to be assigned into all paths and in all possible sequences in each path is evaluated. In each path, a sequence that gives the minimum additional service time is selected. This selected sequence then compared among other selected sequences from all possible path. Finally, the vertex is assigned into a path that gives the minimum additional time needed and does not exceed T_{max} . The algorithm of this decoding stage is presented below:

1. For each vertex ($i = 1$ to $N-2$)
2. For each path ($j = 1$ to P)
3. For each sequence ($l = 1$ to $c_j + 1$)
4. Insert vertex i to sequence l in path j ($S_{jli} = v_{pi}$).
5. Calculate the new total time needed for path j (t_{ij})
6. Update the best new total time for path j (t_{bj}), and its sequence (s_{bji})
7. Repeat step 4 until $l = c_j + 1$
8. If the best new total time for path j (t_{bj}) is not exceed T_{max} , calculate the additional time for path j ($d_j = t_{bj} - t_j$). Else, set d_j as big number (i.e., as T_{max})
9. Repeat step 3 until the last path.
10. Compare the additional time for each path and chose

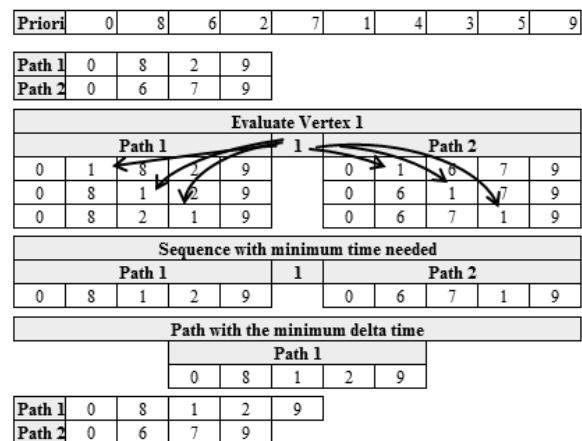


Figure 3. Illustration of decoding method.

the path that has minimum additional time and the time needed is not exceed T_{max} .

11. Update the sequence (s_{ji}) and time (t_j) for the chosen path.
12. Repeat step 2 until all vertex is evaluated.

where, i is vertex priority index, N is number of vertices, j is path index, P is number of paths, l is sequence index, s_{ji} is vertex at i^{th} sequence in path j , t_j is time needed of path j , c_j is number of vertex assigned in path j (excluded start and end point), s_{ijl} is temporary vertex at l^{th} sequence in path j , v_{pi} is i^{th} priority of vertex, t_{ij} is temporary time needed of path j , s_{bjl} is best vertex at l^{th} sequence in path j , t_{bj} is temporary time needed of path j , and d_j is additional time needed of path j . Figure 3 illustrates the decoding method.

4. COMPUTATIONAL RESULT

Computational experiments are conducted by applying the proposed algorithm for solving some benchmark dataset of TOP. As mentioned before, the case problems are the benchmark datasets number 4 to 7 of Chao *et al.* (1996b). The characteristic of the datasets are presented in Table 2.

Each set has three groups (2, 3, and 4), which indicates number of paths in each case. The alphabet index indicates the T_{max} of the case. The smallest index will result in the smallest T_{max} . For example, case p7.4.a. has 102 vertices, 4 paths, and the smallest T_{max} in the group 7.4, which is equal to 5. In contrast, case p7.4.j has 102 vertices, 4 paths, and T_{max} of 50.

Table 2. Characteristic of data set 4–7

Set	Group	Index	No. of case	No. of vertices	No. of paths
4	4.2.	a to t	20	100	2
	4.3.	a to t	20	100	3
	4.4.	a to t	20	100	4
5	5.2.	a to z	26	66	2
	5.3.	a to z	26	66	3
	5.4.	a to z	26	66	4
6	6.2.	a to n	14	64	2
	6.3.	a to n	14	64	3
	6.4.	a to n	14	64	4
7	7.2.	a to t	20	102	2
	7.3.	a to t	20	102	3
	7.4.	a to t	20	102	4

The algorithm is implemented in C# language using Sharp Develop 2.2 in a notebook with Intel Centrino Duo T2250 @1.73 GHz 2.5 GB RAM. PSO computational library called ET-Lib (Nguyen *et al.*, 2010) is used here for expediting the implementation phase. For each case, 10 replications are tried. The PSO parameters are set based on the result of some preliminary experiments that carried out to observe the behavior of algorithm in different parameter setting. The PSO parameters are summarized in Table 3. It is noted that these parameters may not be the best parameter value set for TOP, however, our preliminary experiments indicate that these setting are able to provide consistent and better results among other settings in the experiments.

Tables 4–6 compare the best obtained solutions for each dataset from the proposed algorithm with the summary of results from Souffriau *et al.* (2010), and the results of DPSO from Muthuswamy and Lam (2011). The results are then summarized in Table 7. For each approaches, the number of cases in which the best known solution are found and the average gap to the best known solution is given, as well as average CPU times in seconds.

This proposed algorithm is able to obtain 65 cases in which the best known solution are found, with average gap of 1.83% from best known solution in 366.0 seconds of average CPU times. The quality of this result is still below the result obtained from SPR, ASe, and SVF, for example, SPR is able to find 131 of best known solution is found, with average gap of 0.06% in 212.4 seconds of average CPU times. It is noted that those algorithms a recombining more than one approach into single algorithm. However, the proposed algorithm gives competitive results compared to other basic algorithm, such as TMH, GLS, and DPSO. The TMH algorithm is able to find only 34 of best known solution, with average gap of 1.47% in 336.6 seconds of average CPU times. The GLS is able to find only 21 of best known solution, with average gap of 2.71% in 7.8 seconds of average CPU times. The DPSO is able to find only 39 of best known solution with average gap of 1.45%.

Table 3. Summary of particle swarm optimization parameters

Parameter	Value
Number of particle	$L = 100$
Number of iteration	$T = 1,000$
Number of neighbor	$K = 5$
First inertia weight	$w(t) = 0.9$
Last inertia weight	$w(T) = 0.4$
Personal best position acceleration constant	$c_p = 2$
Global best position acceleration constant	$c_g = 2$

Table 6. Results of data set 7

Case	Best	TMH	GTP	GTF	FVF	SVF	GLS	ASe	ADC	ARC	ASi	SVNS	FPR	SPR	DPSO	CPSO
p7.2.d	190	190	190	190	190	190	190	190	190	190	190	182	190	190	190	190
p7.2.e	290	290	290	290	289	290	279	290	290	290	290	289	290	290	290	290
p7.2.f	387	382	387	387	387	387	340	387	387	387	387	387	387	387	387	387
p7.2.g	459	459	456	459	459	459	440	459	459	459	459	457	459	459	459	459
p7.2.h	521	521	520	520	521	521	517	521	521	521	521	521	521	521	521	520
p7.2.i	580	578	579	579	575	579	568	580	579	579	579	579	578	580	578	577
p7.2.j	646	638	643	644	643	644	633	646	646	646	646	632	646	646	623	633
p7.2.k	705	702	702	705	704	705	691	705	704	704	704	700	702	705	698	694
p7.2.l	767	767	758	767	759	767	748	767	767	767	767	758	759	767	755	745
p7.2.m	827	817	827	824	824	827	798	827	827	827	827	827	816	827	807	806
p7.2.n	888	864	884	888	883	888	861	888	878	878	878	866	888	888	868	865
p7.2.o	945	914	933	945	945	945	897	945	945	940	941	928	932	945	908	927
p7.2.p	1002	987	1000	1002	1002	1002	954	1002	991	993	993	955	993	1002	970	980
p7.2.q	1044	1017	1041	1043	1038	1044	1031	1043	1042	1043	1043	1029	1043	1044	1020	1005
p7.2.r	1094	1067	1091	1088	1094	1094	1075	1094	1093	1088	1094	1069	1076	1094	1066	1060
p7.2.s	1136	1116	1123	1128	1136	1136	1102	1136	1136	1134	1131	1118	1125	1136	1097	1070
p7.2.t	1179	1165	1172	1174	1168	1179	1142	1179	1179	1179	1179	1154	1168	1175	1129	1116
p7.3.h	425	416	425	425	425	425	418	425	425	425	425	425	425	425	419	425
p7.3.i	487	481	487	487	487	487	480	487	487	486	487	480	485	487	485	487
p7.3.j	564	563	564	564	562	564	539	564	564	564	564	543	560	564	557	562
p7.3.k	633	632	633	633	632	633	586	633	632	633	633	633	633	633	633	629
p7.3.l	684	681	683	679	681	681	668	684	683	684	684	681	684	684	681	674
p7.3.m	762	756	749	755	745	762	735	762	762	762	762	743	762	762	754	762
p7.3.n	820	789	810	811	814	820	789	820	819	819	820	804	813	820	813	789
p7.3.o	874	874	873	865	871	874	833	874	874	874	874	841	859	874	848	849
p7.3.p	929	922	917	923	926	927	912	929	925	926	925	918	925	927	919	885
p7.3.q	987	966	976	987	978	987	945	987	987	987	987	966	970	987	960	967
p7.3.r	1026	1011	1018	1022	1024	1022	1015	1026	1024	1021	1022	1009	1017	1021	1017	963
p7.3.s	1081	1061	1081	1081	1079	1079	1054	1081	1081	1081	1077	1070	1076	1081	1064	997
p7.3.t	1118	1098	1114	1116	1112	1115	1080	1118	1117	1103	1117	1109	1111	1118	1093	1026
p7.4.g	217	217	217	217	217	217	209	217	217	217	217	217	217	217	211	217
p7.4.h	285	285	285	285	285	285	285	285	285	285	285	283	285	285	283	285
p7.4.i	366	359	366	366	366	366	359	366	366	366	366	364	366	366	349	366
p7.4.k	520	503	520	520	518	520	511	520	520	520	520	518	518	518	516	519
p7.4.l	590	576	590	588	588	590	573	590	590	590	590	575	581	590	578	590
p7.4.m	646	643	644	646	646	646	638	646	644	646	646	639	646	646	633	646
p7.4.n	730	726	723	721	715	730	698	730	725	725	726	723	723	730	712	730
p7.4.o	781	776	772	778	770	781	761	781	778	781	778	778	780	780	776	763
p7.4.p	846	832	841	839	846	846	803	846	846	838	842	841	842	846	819	825
p7.4.q	909	905	902	898	899	906	899	909	909	909	909	896	902	907	890	904
p7.4.r	970	966	970	969	970	970	937	970	970	970	970	964	961	970	948	964
p7.4.s	1022	1019	1021	1020	1021	1022	1005	1022	1019	1021	1019	1019	1022	1022	988	1008
p7.4.t	1077	1067	1071	1071	1077	1077	1020	1077	1072	1077	1077	1073	1066	1077	1066	1051

TMH: Tabu search heuristic, GTP: Tabu search with penalty strategy, GTF: Tabu search with feasible strategy, FVF: fast variable neighborhood search, SVF: slow variable neighborhood search, ASe: ant colony sequential, ADC: ant colony deterministic-concurrent, ARC: ant colony random-concurrent, ASi: ant colony simultaneous, GLS: guided local search, SVNS: skewed variable neighborhood search, FPR: fast path relinking, SPR: slow path relinking, DPSO: discrete particle swarm optimization, CPSO: continuous PSO.

Table 7. Summary of results

	TMH	GTP	GTF	FVF	SVF	GLS	ASe	ADC	ARC	ASi	SVNS	FPR	SPR	DPSO	CPSO
#Best	34	69	94	97	128	21	130	80	81	84	44	78	131	39	65
Avg. gap (%)	1.47	0.54	0.25	0.22	0.06	2.71	0.11	0.42	0.47	0.39	1.08	0.46	0.06	1.45	1.83
Avg. CPU (s)	336.6	100.0	146.6	18.9	268.6	7.8	252.3	213.8	204.8	215.0	3.8	5.0	212.4	-	366.0

TMH: Tabu search heuristic, GTP: Tabu search with penalty strategy, GTF: Tabu search with feasible strategy, FVF: fast variable neighborhood search, SVF: slow variable neighborhood search, ASe: ant colony sequential, ADC: ant colony deterministic-concurrent, ARC: ant colony random-concurrent, ASi: ant colony simultaneous, GLS: guided local search, SVNS: skewed variable neighborhood search, FPR: fast path relinking, SPR: slow path relinking, DPSO: discrete particle swarm optimization, CPSO: continuous PSO.

5. CONCLUSION AND FURTHER STUDY

This paper proposed a new approach to solving TOP by using continuous PSO. The particle's position is decoded into an order of vertex's priority, which will be evaluated in each sequence in each path. Finally, the vertex will be assigned in a path that gives the minimum delta path and does not exceed T_{max} .

The result of this proposed algorithm is not as good as the best approach proposed in the past. The best approach, SPR, gave the minimum average gap from best known result, and the most best known solution is found.

However, this paper concludes that continuous PSO algorithm can solve TOP with competitive result, especially for simple cases of TOP. This continuous PSO also gives better number of best solution found than other approaches that used basic algorithm, such as TMH, GLS, and DPSO.

Further research can be conducted to improve the performance of this proposed algorithm, such as parameter optimization and decoding alternatives. Although the PSO parameter set used in this paper came from some preliminary experiments, it may not be the best one. 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.

ACKNOWLEDGMENTS

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.

REFERENCES

- Ai, T. J. and Kachitvichyanukul, V. (2009), A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery, *Computers & Operations Research*, **36**(5), 1693-1702.
- Archetti, C., Hertz, A., and Speranza, M. (2007), Metaheuristic for the team orienteering problem, *Journal of Heuristics*, **13**(1), 49-76.
- Butt, S. E. and Cavalier, T. M. (1994), A heuristic for the multiple tour maximum collection problem, *Computers & Operations Research*, **21**(1), 101-111.
- Chang, B. C. H., Ratnaweera, A., Halgamuge, S. K., and Watson, H. C. (2004), Particle swarm optimisation for protein motif discovery, *Genetic Programming and Evolvable Machines*, **5**(2), 203-214.
- Chao, I., Golden, B. L., and Wasil, E. A. (1996a), The team orienteering problem, *European Journal of Operational Research*, **88**(3), 464-474.
- Chao, I., Golden, B. L., and Wasil, E. A. (1996b), A fast and effective heuristic for the orienteering problem, *European Journal of Operational Research*, **88**(3), 475-489.
- Clerc, M. (2004), Discrete particle swarm optimization, illustrated by the traveling salesman problem. In: *New Optimization Techniques in Engineering*, Springer, Berlin, Germany, 219-239.
- Hu, X. (2006), Particle swarm optimization, cited 2013 Sep 1, Available from: <http://www.swarmintelligence.org>.
- Ke, L., Archetti, C., and Feng, Z. (2008), Ants can solve the team orienteering problem, *Computers & Industrial Engineering*, **54**(3), 648-665.
- Muthuswamy, S. and Lam, S. S. (2011), Discrete particle swarm optimization for the team orienteering problem, *Memetic Computing*, **3**(4), 287-303.
- Nguyen, S., Ai, T. J., and Kachitvichyanukul, V. (2010), *Object Library for Evolutionary Techniques ET-Lib: User's Guide*, High Performance Computing Group, Asian Institute of Technology, Thailand.
- Souffriau, W., Vansteenwegen, P., Berghe, G. V., and Van Oudheusden, D. (2010), A path relinking approach for the team orienteering problem, *Computers & Operations Research*, **37**(11), 1853-1859.
- Souffriau, W., Vansteenwegen, P., Vertommen, J., Berghe, G. V., and Van Oudheusden, D. (2008), A personalized tourist trip design algorithm for mobile tourist guides, *Applied Artificial Intelligence*, **22**(10), 964-985.
- Tang, H. and Miller-Hooks, E. (2005), A tabu search heuristic for the team orienteering problem, *Computers & Operations Research*, **32**(6), 1379-1407.
- Tsiligirides, T. (1984), Heuristic methods applied to orienteering, *Journal of the Operational Research Society*, **35**(9), 797-809.
- Vansteenwegen, P., Souffriau, W., Berghe, G. V., and Van Oudheusden, D. (2009a), A guided local search metaheuristic for the team orienteering problem, *European Journal of Operational Research*, **196**(1), 118-127.
- Vansteenwegen, P., Souffriau, W., Berghe, G. V., and Van Oudheusden, D. (2009b), Metaheuristics for tourist trip planning. In: *Metaheuristics in the Service Industry*, Springer, Berlin, Germany, 15-31.
- Vansteenwegen, P., Souffriau, W., and Van Oudheusden, D. (2011), The orienteering problem: a survey, *European Journal of Operational Research*, **209**(1), 1-10.