TÜBİTAK

Turk J Elec Eng & Comp Sci, Vol.20, No.2, 2012, © TÜBİTAK
doi:10.3906/elk-1101-1008

Discrete particle swarm optimization for the team orienteering problem

Aişe Zülal ŞEVKLİ[1], Fatih Erdoğan SEVİLGEN[2,*]
[1]*Department of Computer Engineering, Faculty of Engineering, Fatih University, Büyükçekmece, İstanbul-TURKEY*
[2]*Department of Computer Engineering, Faculty of Engineering, Gebze Institute of Technology, Gebze, Kocaeli-TURKEY*
e-mail: sevilgen@bilmuh.gyte.edu.tr

Received: 01.02.2011

Abstract

In this paper, a novel discrete particle swarm optimization (PSO) algorithm is proposed to solve the team orienteering problem (TOP). Discrete evaluation is achieved by redefining all operators and operands used in PSO. To obtain better results, a strengthened PSO, which improves both exploration and exploitation during the search process, is employed. Our algorithm achieves the best known solutions in a short time compared to previous heuristics for the TOP.

Key Words: *Particle swarm optimization, reduced variable neighborhood search, team orienteering problem, vehicle routing problem with profits*

1. Introduction

The team orienteering problem (TOP) is a subset selection version of the well-known vehicle routing problem with profits. The objective of the TOP is to construct a certain number of paths starting at an origin and ending at a destination that maximizes the total profit without violating prescribed limits. The problem is inspired from and named after an outdoor sport usually played on mountains or in forested areas.

The TOP can be described in detail as follows: Let $G = (N, E)$ be a graph where N denotes the set of nodes (control points) and E denotes the set of edges between the nodes in N. Each node n_i in N $(1 \leq i \leq k)$ has a score $s_i \geq 0$; the scores of n_1 (the origin) and n_k (the destination) are 0, such that $s_1 = s_k = 0$. Each edge between n_i and n_j has a cost, d_{ij}, associated with it. There is a set of m vehicles located at n_1. The objective of the TOP is to find a path for each vehicle. Each path should start at n_1, end at n_k, and maximize the total profit by satisfying a cost constraint (e.g., the total cost of the edges on each path should be less than a specified limit, T_{max}). Because of the limitation, each vehicle cannot visit all control points. Golden et al.

*Corresponding author: Department of Computer Engineering, Faculty of Engineering, Gebze Institute of Technology, Gebze, Kocaeli-TURKEY

[1] proved that the orienteering problem, which is a single-vehicle version of the TOP, is NP-hard. A linear programming model for the TOP can be found in [2].

In this paper, we consider the Euclidean TOP, where the nodes are on the Euclidean plane. In such a problem, the graph is a full graph and the score of an edge is the Euclidean distance between 2 nodes adjacent to the edge.

In the literature, exact methods [3,4] and several metaheuristics, including population-based methods such as memetic algorithm [5], ant colony optimization, [6] trajectory-based methods such as variable neighborhood search [7], tabu search [7,8], and guided local search [9], have been explored for the TOP.

In this paper, we present a discrete strengthened particle swarm optimization (DStPSO) algorithm based on strengthened particle swarm optimization (StPSO) to solve the TOP. We proposed StPSO as a variation of particle swarm optimization (PSO) in [10]. PSO was initially proposed by Eberhart and Kennedy for solving continuous optimization problems [11]. Later, it was also applied to combinatorial optimization problems [12-14]. Among other metaheuristic methods, PSO is popular for its ease of application and fast convergence to a near optimal solution.

We employed StPSO for both continuous and discrete optimization problems in [10] and demonstrated that StPSO improves both the exploration and exploitation of PSO. The results obtained by using StPSO are better for both continuous and discrete problems. However, the improvement for discrete problems is limited due to the simple adaptation of StPSO. Continuous search operation, including all continuous operators and components of the StPSO, is preserved. To transform the continuous search space to a discrete problem space, an extra conversion process is used. Because of incongruities between the 2 domains, the performance of the search operation diminishes.

On the other hand, in DStPSO, we redefine all operators and components in StPSO for the TOP. Although the general flow of StPSO is preserved in DStPSO, the search operation is performed on a natural discrete search space using discrete operators.

DStPSO was tested using 138 benchmark problems. It achieved the best known solutions for all instances within a reasonable amount of time. The results demonstrate that our algorithm is a promising alternative not only for the TOP but also for similar discrete problems.

2. Particle swarm optimization

PSO is a population-based method in which a swarm includes n individuals called particles. Each particle has a d-dimensional position vector representing a candidate solution and a d-dimensional velocity vector expressing the current tendency of the particle during its search journey. The initial swarm can be constructed randomly or by using predetermined values. At each step, the velocity of each particle is reevaluated based on the particle's inertia as well as its social interaction (the swarm's experience) and personal experience. The experience of each particle is usually captured by its best position so far (*pbest*). The experience of the swarm is captured by the global best position (*gbest*) obtained by the swarm. In the course of several iterations, particles make use of this experience and are supposed to move toward the optimum position.

The pseudocode of the standard PSO algorithm is illustrated in Figure 1. Optimization is achieved by several iterations of update-evaluate steps. During the update step (line 10), the velocity and the position vector of each particle are calculated by using Eqs. (1) and (2). In these equations, $v_{i,j}^t$ and $p_{i,j}^t$ are the velocity

and the position of the ith particle ($1 \leq i \leq n$) in the jth dimension ($1 \leq j \leq d$) at iteration t, respectively. Parameter w is the inertia weight. The inertia weight serves as a balancing factor between exploration and exploitation. Large inertia weight at the beginning of the search enables more exploration, while smaller inertia weight facilitates more exploitation later. The parameters c_1 and c_2 are learning factors, which are the weights for contributions of personal experience and social interaction. The stochastic behavior of PSO is achieved by random numbers r_1 and r_2, which are positive numbers generally uniformly distributed in [0, 1].

$$v_{i,j}^t = wv_{i,j}^{t-1} + c_1 r_1 (pbest_{i,j}^{t-1} - p_{i,j}^{t-1}) + c_2 r_2 (gbest_j^{t-1} - p_{i,j}^{t-1}) \tag{1}$$

$$p_{i,j}^t = p_{i,j}^{t-1} + v_{i,j}^{t-1} \tag{2}$$

After the update step, the fitness function value is calculated (line 11) for each particle based on its position (the candidate solution represented by the particle.) The local best position, *pbest*, of each particle (line 12) and the global best position, *gbest*, of the swarm (line 13) are updated if the candidate solution is better than *pbest* or *gbest*, respectively. The stopping condition (line 14) of the update-evaluate iterations is usually the attainment of a maximum number of iterations or a maximum number of iterations between 2 improvements.

```
 1 Procedure PSO
 2  Initialize Parameters
 3  Initialize Population
 4  For each particle
 5   Evaluate
 6   Update local best (pbest)
 7  Update global best (gbest)
 8  Do
 9   For each particle
10     Update velocity and position
11     Evaluate
12     Update local best
13   Update global best
14  While (Not Terminated)
15 End Procedure
```

Figure 1. Pseudocode of the standard PSO algorithm.

2.1. Strengthened PSO

Since PSO was introduced, it has been adapted to solve several optimization problems. Meanwhile, some shortcomings like premature convergence or lack of intensification around the local best locations were observed. To improve search efficiency and rectify these deficiencies in the standard algorithm, researchers have proposed several modifications to PSO.

One of the latest improved PSO versions is StPSO. The pseudocode of the StPSO algorithm is illustrated in Figure 2. The main focus of StPSO is on pioneering particles, which achieve or enhance the swarm's experience. These particles are assumed to be either converged or potentially converged. They are processed in 2 steps. First, an external local search is initiated for each pioneering particle (line 11). After the local search, only the local experience of each pioneering particle is updated. Swarm experience is not changed. This step strengthens the exploitation mechanism in PSO. Second, at the same iteration, the random moving strategy

is performed on each pioneering particle in order to force the particle to continue exploration with its past experience (line 13). In this way, the exploration mechanism of PSO is improved and premature convergence is largely avoided. The other particles continue to search the solution space as in the standard PSO algorithm.

```
 1 Procedure StPSO
 2  Initialize Parameters
 3  Initialize Population
 4  For each particle
 5    Evaluate
 6    Update local best (pbest)
 7  Update global best (gbest)
 8  Do
 9   For each particle
10    If (fitness = fitness of gbest)
11         position ← LS(position)
12         Update local best
13         Assign new random position
14    Else
15         Update velocity
16         Update position
17    Evaluate
18    Update local best
19   Update global best
20  While (Not Terminated)
21 End Procedure
```

Figure 2. Pseudocode of the StPSO algorithm.

3. Discrete StPSO for TOP

In DStPSO, the position and velocity vector and all continuous operators in PSO are redefined for discrete processing in the discrete solution space of the TOP.

Both the position and velocity of a particle are represented as lists of control points. The position includes m lists representing a candidate TOP solution. Note that m is the number of vehicles and each list denotes a path for the corresponding vehicle. Each path starts from the starting point, visits all control points in the corresponding position list, and ends at the destination point. The solution is always feasible; in other words, the total cost of the edges on the path is less than T_{max}. The velocity is a list including the control points that are not in any paths of the same particle.

The DStPSO algorithm preserves the general flow of the StPSO, which is illustrated in Figure 2. The algorithm starts with the initialization of the particles in the swarm. The position and velocity of a particle are initialized by using a random permutation of control points. To obtain position lists, starting from the first control point in the permutation, the control points are inserted into the first list between the starting point and the destination point, one by one, until the prescribed cost limit is exceeded. This process is repeated for the other $m-1$ lists using the remainder of the permutation. The other control points that are not in the position lists are inserted into the velocity list while preserving the permutation order.

3.1. Discrete operators

In DStPSO, the velocity and position update equations used in standard PSO are redefined as follows.

$$v_i^t = \left[w_i \otimes v_i^{t-1}\right] \oplus \left[(c_1 \otimes (pbest_i^{t-1} \Theta\, p_i^{t-1})) \oplus (c_2 \otimes (gbest^{t-1} \Theta\, p_i^{t-1}))\right] \tag{3}$$

$$p_i^t = p_{i,}^{t-1} \circ v_i^t \tag{4}$$

$$v_i^t = p_{all}\, \Theta\, p_i^t \tag{5}$$

In Eqs. (3) and (4), the velocity and position are calculated using new operators, respectively. For the next iteration, the velocity of the particle is reevaluated in Eq. (5) by using p_{all} and the new position. p_{all} is a random permutation of the control points. The new operators used in the above equations are explained below.

Θ operator: Let p_1 and p_2 be 2 positions; then $(p_1 \Theta p_2)$ is a velocity. The velocity list includes the control points that are in one of the lists in p_1 but not in any list in p_2. If p_1 is *gbest*, the velocity list has high priority (hp); if p_1 is *pbest*, it has low priority (lp). Otherwise, it has no priority (np).

$\otimes$ operator: The left operand is a coefficient ($0 < c < 1$) and the right operand is a velocity (v). The result ($c \otimes v$) is a velocity. The coefficient represents the probability of a control point in v stays in the resulting velocity list. The result can be calculated as follows: starting from the first control point in v, a random number that is uniformly distributed in the range of [0, 1] is generated. If the random number is less than c, this control point is appended to the new velocity list. The operation does not alter the priority attribute.

$\oplus$ operator: This operator combines 2 velocity lists. Remember that a velocity list includes a priority attribute. The operation is performed based on the priority values of its operands. If an operand has high priority and the other has low priority, the result ($v_{hp} \oplus\ v_{lp}$) is obtained by placing the control points that are in the intersection of these lists at the beginning, and then placing the remaining control points in v_{hp} and in v_{lp} in order. If one of the operands has no priority, the result ($v_{hp/lp} \oplus v_{np}$) includes the control points in $v_{hp/lp}$ at the beginning and the remaining control points in v_{np} at the end.

$\circ$ operator: This operator takes a position and a velocity list as operands. The velocity includes the control points to be inserted into the position lists. We apply *node insert for increasing profit*, explained in the next section, to the position lists for each control point in the velocity list, one by one.

3.2. Local search

Reduced variable neighborhood search (RVNS) is employed in DStPSO as the local search method. RVNS is a variation of variable neighborhood search (VNS), which was introduced as an optimization method for combinatorial optimization problems [15]. In RVNS, the solution space is searched with a systematic change of neighborhood. The details of the algorithm are presented in [10]. In the following text, the neighborhood structures are explained in the same order as they are used in RVNS.

3.2.1. Node insert for increasing profit

In this structure, a randomly selected control point in the velocity list is inserted into a randomly selected position list. The control point is inserted into the cheapest location, which minimizes the cost function (cheapest insertion method). If the list still satisfies the cost constraint, a new solution in the neighborhood is obtained. Otherwise, the solution is not feasible. To find a feasible solution, a control point that has a lower score than

the inserted control point and minimizes the path cost when it is removed is selected. If removal of the control point makes the solution feasible (respects the cost constraint), the control point is removed and a new solution in the neighborhood is obtained after one insertion and one deletion. Either way, a feasible solution with more profit than the old one is generated and the RVNS continues to search within this neighborhood. If a feasible solution cannot be obtained, the RVNS switches to the following neighborhood.

3.2.2. Node insert for decreasing cost

The aim of this structure is to decrease the cost of a TOP solution. To realize this aim, 2 random position lists are selected from the solution. These 2 lists could be the same list. In such a case, a control point chosen randomly from the position list is reinserted in the same list using the cheapest insertion method. Otherwise, a 1-way control-point move or a 2-way interchange is performed between the lists. A control point chosen randomly from the first position list is inserted into the second list using the cheapest insertion method. If the second list does not satisfy the feasibility constraint after this 1-way move, a random control point from the second list is inserted into the first list using the cheapest insertion method to complete the 2-way interchange.

At the end of the operation, a new solution is generated in the neighborhood if the feasibility of the position is preserved and at least one of the position lists' cost is reduced.

If a successful neighbor is generated, the RVNS continues to search within the first neighborhood. If a THRESHOLD-times (back-to-back) unimproved solution is observed from the first 2 neighborhoods, the RVNS switches to the following last neighborhood.

3.2.3. Path invert

This neighborhood is also called a 2-opt move. To solve travelling salesman-based problems, the 2-opt local search algorithm is frequently used in order to decrease the length of a tour. In a 2-opt move, if the tour is crossing over itself, 2 crossing edges are replaced with noncrossing edges to obtain cost reduction. In fact, the 2-opt move is the same as inverting a subsequence of control points in a position list. Therefore, we call this neighborhood a path invert. The path invert is employed on the position lists many times until either the cost is reduced or all pairs of edges in the list are tested.

Note that the location of the starting and ending control points are not altered while producing a new TOP solution in any of the neighborhoods above.

4. Experimental results

To observe the effects of the proposed algorithm, we performed experimental analysis on DStPSO, its 2 subvariants, and discrete standard PSO (DPSO). One subvariant that only performs a random moving strategy on the pioneering particle without performing the local search is called discrete diversification strengthened PSO (DDS-PSO). Particles in DDS-PSO tend to perform more exploration than exploitation since they are not allowed to stay near good positions. The other subvariant, which only conducts RVNS for pioneering particles, is called discrete intensification strengthened PSO (DIS-PSO).

All PSO algorithms were tested on 138 benchmark problems. Problems were categorized into 3 datasets containing 32, 21, and 33 control points [2]. Each dataset included problems with 2, 3, and 4 vehicles and various T_{max} values.

Experiments were performed on an Intel P4 2.8 GHz PC with 1 GB of memory. In the experimentation, the swarm size was 10 particles. The parameter values were $c_1 = c_2 = 0.5$, and the initial value of inertia weight w for all particles was 0.7. After each iteration, inertia weight decreased by 2%. Runs were terminated when *gbest* had no improvement for 300 consecutive times. For RVNS, the THRESHOLD parameter value was 10 and the stopping condition was 20 back-to-back iterations without improvement. The parameter values were determined experimentally.

Each problem was run 10 times and the results were compared based on the computation time (CPU), relative percentage error (RPE), and standard deviation (Std. Dev.) of the repetitions. RPE indicates whether an algorithm finds the best known solution throughout the repetitions. Performances of the 4 PSO versions for the datasets are given in Table 1.

Table 1. Comparison of various PSO applications: the average results for datasets 1, 2, and 3 based on RPE, Std. Dev., and CPU time.

Dataset	# of vehicles	CPU (s)				RPE				Std. Dev.			
		DPSO	DDS-PSO	DIS-PSO	DStPSO	DPSO	DDS-PSO	DIS-PSO	DStPSO	DPSO	DDS-PSO	DIS-PSO	DStPSO
1	2	1.1	4.3	3.5	1.3	0.3	2.3	0.0	0.0	1.9	2.3	0.9	0.0
	3	1.3	4.0	1.7	0.7	0.5	2.6	0.0	0.0	2.0	2.6	1.3	0.7
	4	0.9	2.6	0.8	0.4	0.3	1.7	0.0	0.0	1.1	1.7	0.6	0.4
Avg.		**1.1**	**3.7**	**2.1**	**0.9**	**0.4**	**1.3**	**0.0**	**0.0**	**1.7**	**2.2**	**0.9**	**0.3**
2	2	0.2	0.3	0.2	0.2	0.2	0.5	0.0	0.0	2.8	3.1	0.0	0.0
	3	0.1	0.2	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.6	0.0	0.0
	4	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Avg.		**0.1**	**0.2**	**0.1**	**0.1**	**0.1**	**0.2**	**0.0**	**0.0**	**1.0**	**1.2**	**0.0**	**0.0**
3	2	2.6	3.6	4.6	3.1	1.2	2.5	0.1	0.0	6.4	6.8	4.5	1.1
	3	2.2	3.2	1.7	1.1	2.0	3.7	0.0	0.0	7.4	7.6	6.0	0.8
	4	1.6	2.6	0.8	0.6	2.4	4.2	0.0	0.0	8.7	12.4	4.4	0.3
Avg.		**2.1**	**3.2**	**2.4**	**1.7**	**1.9**	**3.4**	**0.0**	**0.0**	**7.5**	**8.8**	**4.9**	**0.7**

Since the algorithms (DPSO, DIS-PSO, and DStPSO), except for DDS-PSO, have inherited or imported exploitation mechanisms, their results were better than those of DDS-PSO. The RPE values for DPSO indicate that the inherited exploitation mechanism is not sufficient to produce best known solutions. DIS-PSO and DStPSO achieved the best known solutions for all problem instances. However, DStPSO outperformed DIS-PSO based on CPU and Std. Dev. measures. Thus, it provided better solutions in less time.

The DStPSO algorithm is compared with the following algorithms in Table 2:

- CGW: Heuristic algorithm [2]
- TMH: Tabu search [8]
- GTP: Tabu search with penalty strategy [7]
- GTF: Tabu search with feasible strategy [7]
- FVF: Fast VNS [7]
- SVF: Slow VNS [7]
- ACO: Ant colony optimization with sequential method [6]
- MA: Memetic algorithm [5]

The average of the RPE values for the 138 benchmark problems in Table 2 indicate that DStPSO achieved the best known solutions for all problems, like many other algorithms. The run-time performance of the DStPSO is competitive with other state-of-the-art algorithms. Note that the experimentations in [5-7] were performed on a similar computer. Both the quality of the solutions and the execution times indicate that DStPSO is a promising technique for solving the TOP.

Table 2. Performance of DStPSO against previous studies.

	# Best	Avg. RPE	Avg. CPU
CGW	112	0.97	11.91
TMH	121	0.42	-
GTP	136	0.05	4.12
GTF	138	0.00	1.22
FVF	138	0.00	0.11
SVF	138	0.00	6.92
ACO	138	0.00	5.45
MA	138	0.00	1.13
DStPSO	138	0.00	1.02

5. Conclusion

In this paper, a novel PSO-based algorithm for solving the TOP was proposed. In our algorithm, new discrete operators were defined and used for discrete evaluation of position and velocity. To achieve better results, we employed StPSO, whose exploration mechanism and exploitation mechanism are enhanced by reinitiating velocities and embedding a local search method, respectively. The performance of DStPSO was examined for 138 benchmark problems. The best known solutions were found in a comparable amount of time for all problems. The results indicate that DStPSO is a competitive method for the TOP. Investigating the applicability of DStPSO for other subset selection type problems is a promising future research direction.

References

[1] B.L. Golden, L. Levy, R. Vohra, "The orienteering problem", Naval Research Logistics, Vol. 34, pp. 307-318, 1987.

[2] I. Chao, B. Golden, E. Wasil, "Theory and methodology – the team orienteering problem", European Journal of Operations Research, Vol. 88, pp. 464-474, 1996.

[3] S. Boussier, D. Feillet, M. Gendreau, "An exact algorithm for team orienteering problems", 4OR, Vol. 5, pp. 211-230, 2007.

[4] S. Butt, D. Ryan, "An optimal solution procedure for the multiple tour maximum collection problem using column generation", Computers & Operations Research, Vol. 26, pp. 427-441, 1999.

[5] H. Bouly, D.C. Dang, A. Moukrim, "A memetic algorithm for the team orienteering problem", Lecture Notes in Computer Science, Vol. 4974, pp. 649-658, 2008.

[6] L. Ke, C. Archetti, Z. Feng, "Ants can solve the team orienteering problem", Computers & Industrial Engineering, Vol. 54, pp. 648-665, 2008.

[7] C. Archetti, A. Hertz, M.G. Speranza, "Metaheuristics for the team orienteering problem", Journal of Heuristics, Vol. 13, pp. 49-76, 2007.

[8] H. Tang, E. Miller Hooks, "A tabu search heuristic for the team orienteering problem", Computers & Operations Research, Vol. 32, pp. 1379-1407, 2005.

[9] P. Vansteenwegen, W. Souffriau, G.V. Berghe, D.V. Oudheusden, "A guided local search metaheuristic for the team orienteering problem", European Journal of Operational Research, Vol. 196, pp. 118-127, 2009.

[10] Z. Sevkli, E. Sevilgen, "StPSO: Strengthened particle swarm optimization", Turkish Journal of Electrical Engineering & Computer Sciences, Vol. 18, pp. 1095-1114, 2010.

[11] R.C. Eberhard, J. Kennedy, "New optimizer using particle swarm theory", Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39-43, 1995.

[12] K. Rameshkumar, R.K. Suresh, K.M. Mohanasundaram, "Discrete particle swarm optimization (DPSO) algorithm for permutation flowshop scheduling to minimize makespan", ICNC, Vol. 3, pp. 572-581, 2005.

[13] C.J. Liao, C.T. Tseng, P. Luarn, "A discrete version of particle swarm optimization for flowshop scheduling problems", Computers & Operations Research, Vol. 34, pp. 3099-3111, 2007.

[14] M.F. Tasgetiren, Y.C. Liang, M. Sevkli, G. Gencyilmaz, "Particle swarm optimization algorithm for makespan and total flowtime minimization in permutation flowshop sequencing problem", European Journal of Operational Research, Vol. 177, pp. 1930-1947, 2007.

[15] N. Mladenovic, P. Hansen, "Variable neighborhood search", Computers & Operations Research, Vol. 24, pp. 1097-1100, 1997.