Solving the Orienteering Problem Using Attractive and Repulsive Particle Swarm Optimization

Herby Dallard and Sarah S. Lam
Systems Science and Industrial Engineering Department
State University of New York at Binghamton
Binghamton, NY 13902
Phone 607-777-4754 | Fax 607-777-4094
herbydallard@yahoo.com
sarahlam@binghamton.edu

Sadan Kulturel-Konak
Management Information Systems
Penn State Berks
Reading, PA 19610
Phone 610-396-6137 | Fax 610-396-6024
sadan@psu.edu

Abstract

The initial study of this research applied the particle swarm optimization (PSO) heuristic to the orienteering problem (OP). PSO is a fairly new evolutionary heuristic-type algorithm created by Drs. Eberhart and Kennedy in 1995. Similar to ant colony optimization, motivation for PSO is nature-based on fish schooling and bees swarming. The OP is a variation of the well-known traveling salesmen problem (TSP) and is an NP-hard benchmark problem. Given a set of nodes with associated scores, the objective of the OP is to find a path that maximizes the total score subject to a given time (or distance) constraint. This paper presents an attractive and repulsive particle swarm optimization (ARPSO), which prevents PSO's weakness of premature convergence by maintaining solution diversity while retaining a rapid convergence. The ARPSO solves the OP with significant improvement in results when compared to PSO and is more competitive to known best published results.

1. Introduction

Orienteering is known as an outdoor sport that originated in Scandinavia as a military exercise. The race form of orienteering usually played in heavily forested areas has a number of "control points" where each point is associated with a score. Each participant has a compass and a map, and is required to visit a subset of the "control points", starting from node *1* and ending at node *n*. The competitors seek to maximize their total scores, while completing the visits within a prescribed amount of time [5]. The competitors who arrive late are either disqualified or charged a severe penalty [5].

The OP is NP-hard and has applications in vehicle routing and production scheduling, as discussed in [5, 7] and can be mathematically formulated as follows [10, 11]:

$$\text{Maximize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} S_i x_{ij} \tag{1}$$

Subject to:

$$\sum_{j=2}^{n} x_{1j} = \sum_{i=1}^{n-1} x_{in} = 1 \tag{2}$$

$$\sum_{i=2}^{n-1} x_{ik} = \sum_{j=2}^{n-1} x_{kj} \le 1, \quad k = 2,\ldots,n-1 \tag{3}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij} \le T_{\max} \tag{4}$$

$$x_{ij} \in \{0,1\}, \quad i,j = 1,\ldots,n \tag{5}$$

Given a set of n nodes with associated scores $S_i \ge 0$, along with start and end nodes (nodes *1* and n) which have no scores, the OP mathematically determines a path that maximizes the total score subject to a given time (or distance) constraint, denoted by T_{max}. Each node has a two

dimensional Cartesian coordinate. A path between nodes *i and j* has an associated cost c_{ij} and each node can be visited at most once. The time (or distance) constraint causes most tours not to include all nodes. The OP is essentially the TSP when the time (or distance) constraint is relaxed just enough to cover all the nodes while the start and end nodes are not specified. The decision variable x_{ij} = 1 represents that the path from node *i* to node *j* is included in the tour from node *1* to node *n*.

There have been various heuristics and evolutionary algorithms [7, 10, 11] applied to the OP. Some of these approaches are found in Chao et al. [3] and Dallard et al. [16]. Additionally, in Chao et al. [3], there are problem sets of specific node size with distance instances (Tmax). Within problem set II (21-node OP), the variations of the distance parameter T_{max} produce a total of 11 test problems with a search space of size 1.2×10^{17}. The problem set II was previously investigated using PSO and we obtained noteworthy results. The reinvestigation in this paper will be to demonstrate the effectiveness of a modified PSO known as attractive repulsive PSO (ARPSO) to solving the OP.

This paper is organized as follows. Section 2 briefly describes the motivation, application, algorithm, and type of search for ARPSO. In section 3, the experimental set-up, results on problem set II along with comparisons to PSO and best known published results are discussed. Finally, section 4 summarizes the conclusions and recommendations for future research.

2. Attractive Repulsive Particle Swarm Optimization

Kennedy and Eberhart developed the PSO in 1995, inspired by some aspect of nature [8, 9]. A good nature analogy of PSO is described as a swarm of birds searching for food where each bird (referred to as a particle) makes use of its own memory as well as knowledge gained by the swarm as a whole to find the best available food source [15]. PSO is related to evolutionary computation and has ties to genetic algorithms, simulated annealing [6], and evolutionary programming, which make it another powerful optimization and probabilistic search algorithm [8]. Additionally, it is easy to understand and implement, and is computationally efficient [9].

The flexibility of PSO in solving various problems is similar to applications traditionally employing genetic algorithms and other evolutionary type algorithms. Therefore, PSO has been applied to combinatorial type problems; for instance, lot sizing problem [12], permutation flow-shop sequence problem [13, 14], and protein motif discovery problem [1].

The ARPSO brings a few modifications on the PSO. Riget and Vesterstrom [17] created the ARPSO to overcome the premature convergence weakness of the PSO. The attraction phase and repulsion phase are two different stages that the model is in when updating velocity of a particle. When the ARPSO is in the attraction phase, it functions exactly as the PSO developed by Kennedy and Eberhart [8]. That is, the velocity update formula operator is the addition. However, when the ARPSO is in the repulsion phase, the velocity update formula operator is subtraction among the velocity variables—this will be described later. Hence, the ARPSO in the attraction phase causes the particles in the swarm to converge towards one another, whereas, repulsion pushes the particles away from one another reproducing initial-like diversification.

Since the ARPSO is an adjustment of the PSO heuristic, it has the same flexibility with the PSO yet is more effective and efficient to solve discrete or continuous optimization problems to which the PSO has been applied. Particularly, Riget and Vesterstrom [17] evaluated ARPSO performance against PSO and genetic algorithm (GA) on four standard n-dimensional multi-modal objective functions (Griewank, AckeyF1, Rosenbrock, and Rastrigin). Their observations were based on two different sets of experiments; (1) standard comparison of performance between the algorithms, and (2) measurement of the percentage of time spent and the percentage of improvements obtained in the two phase attraction and repulsion in ARPSO. Riget and Vesterstrom [17] state that they made experiments with each test function in 20, 50, and 100 dimensions for 50 replications. They conclude: "the ARPSO on all four test problems performs extremely well."…"On truly multi-modal test problems, the ARPSO algorithm by far outperforms the basic PSO and the implemented GA."

2.1. The ARPSO Algorithm

Sharing many similarities with the PSO, the ARPSO algorithm begins with a random initialization of a population of individual particles in the search space once the swarm size has been quantified. According to Clerc [4] each particle has the following features: (1) it has a position and a velocity, (2) it knows its position and the objective function value for this position, (3) it knows its neighbors, the best previous position and objective function value (variant: current position and objective function value), and (4) it remembers its best previous position. At each time cycle, the behavior of a given particle is a compromise between three possible choices: (1) to follow its own way – how much the particle trusts itself now, (2) to go towards its best previous position – how much it trusts its experience, and (3) to go towards the best neighbor's best previous position, or towards the best neighbor (variant) – how much it trusts its neighbors [4]. Once this is completed, the global best of all particles, neighborhood best, and the local best "thus far" of each particle is calculated. The aforementioned compromise can

be formulated as the revised modified equation of the classical PSO [4, 17, 18] for discrete problems:

$$v_{t+1} = wv_t + dir[r_1 c_1 (p_{l,t} - x_t) + r_2 c_2 (p_{g,t} - x_t)\ ] \quad (6)$$

$$x_{t+1} = x_t \Leftarrow InsertSwap_{vi,ei}[p_{nb,t}(v_{t+1}), _{i=rand\,(\mathrm{int})[2,21]}] \quad (7)$$

$$diversity(S) = \frac{1}{|S| * |L|} \sum_{i=1}^{|S|} \sqrt{\sum_{j=1}^{N} \left(p_{lb,i,j,t} - x_{i,j,t}\right)} \quad (8)$$

Where v_t and x_t are velocity and position of some particle at time step t, respectively. Parameters $p_{l,t}$, $p_{nb,t}$, and $p_{g,t}$ are the best local position per particle, best neighbor's current position per particle's neighborhood, and best global position in the entire swarm at time step t, respectively. The learning parameters w, c_1, c_2 are the cognitive and social factors which are chosen as w [0, 1] and c_1, c_2 [0, 2]. Variables r_1 and r_2 are random constants on the interval [0, 1] drawn at each time step t for each particle. Most implementations of the PSO usually have w equal to one or each particle always trusts itself. Variable *dir* directs the velocity of a swarm being updated by attraction "addition (+1)" or repulsion "subtraction (-1)." The current direction and the diversity value of the swarm greater or lower than a threshold value [17] determine if the swarm will be updated in an attraction or repulsion manner. Variable S symbolizes the swarm at discrete epochs of time. Parameter $|S|$ and $|L|$ represent the swarm size and the length of the longest diagonal in the search space, N is the dimensionality of the problem, $p_{1b,i,j,t}$ is the j^{th} value of the i^{th} best previous position per particle at time step t, and $x_{i,j,t}$ is the j^{th} value of the i^{th} particle at time step t.

2.2. Solution Methodology

An ARPSO algorithm was implemented to solve the OP as outlined in section 2.1. In our original PSO investigation [16] the velocity vectors were computed similar to what was proposed in [4] over the entire particle position. However, this method obtained mediocre results because the updates moved each particle very far in the search space at each time step. This can be likened to a particle being in the neighborhood of a near optimal solution but instead of further investigation of other solutions in that neighborhood over the next few time steps, it moves out of the neighborhood in the next time step and out of that neighborhood in the following time step and etc. This is not very efficient. Consequently, we employed a velocity update scheme successfully utilized by Hu et al. [18] as an alternative to updating the velocity of the entire particle. Instead of the velocity representing the entire particle position; each permutated element in the particle would have its own velocity. Specifically, for the 21-node OP, each particle can have at most 21 elements, ranging from 1 to 21--without duplication--and each element would have its own velocity. Since the 21-nodes are discrete, the velocity elements would be restricted to absolute integer values. Therefore, when equation (6) performs the subtraction of the current particle position (x_t) from the local best particle ($p_{l,t}$) or global best particle ($p_{g,t}$), the mathematical distance formula for each element between each particle is computed and rounded to an integer value. Next, the velocity vector (v_{t+1}) of a particle is transformed to have a normalized representation. When normalization is accomplished, the new normalized velocity vector is further transformed into a cumulative normalized velocity vector. This essentially ends the velocity update process in equation (6) and begins the particle update procedure in equation (7). A random number between 0 and 1 is taken. The velocity index i_v of the element greater than or equal to that random number is obtained with reference to the cumulative normalized velocity vector. The index i_v value of that velocity element is used to obtain the value of the element in $p_{nb,t}$ (best neighbor position of particle) with having index i_v. This constant element value of $p_{nb,t}$ (i_v) equals to C is then searched for in the current particle x_t and is found at an index i_x. If index value i_x is not equal to index value i_v, the two values are swapped in x_t ($swap[x_t(i_v),x_t(i_x)]$). If both index i_v and i_x are found equal, a form of mutation occurs where two integer elements in particle x_t between 2 and 21 are randomly generated and swapped in x_t. These minuscule changes in the particle position are incremental step size changes in a particle rather than numerous changes throughout the entire particle. That is, over several time step t, a particle investigates the solutions in its neighborhood space before possibly moving to new neighborhoods and finally the region of swarm convergence in the search space. This method of velocity update described is more sensible and is anticipated to provide a more effective exploration of the search space than the velocity update method of [4, 16] used for PSO. An illustration of our previously discussed velocity and position update is shown in Table 1. In Table 1, we assume that all parameter constants in equation (6) are equal to 1.

As mentioned previously, each particle has a tour always starting at node 1 and terminating at node 21. The nodes visited in between those two nodes have to yield a feasible tour by having a total Euclidean distance calculated to be less than or equal to T_{max}. Thus each node has some x and y coordinate positions along with an associated score value. After each velocity update and position update of all particles in the swarm, the feasibility of each particles tour is verified and corrected by shrinking it (deleting nodes if necessary) to yield a feasible tour with

respect to T_{max}. Immediately afterwards the deleted elemental nodes in the array are rearranged appropriately to expand the particle beyond its feasible path and maintain 21 unique nodes per particle. An example of this is shown in Table 2.

Once the feasibility, shrinking and expansion are completed, the objective function value for that tour is computed to obtain the total tour score associated with all the feasible nodes visited. Next, the local best tour for that particle and the neighborhood of best particle are updated and stored, if necessary. All of the aforementioned procedures are repeated for each particle in the swarm and then the global best tour value of all particles during that time step is updated.

Table 1. Update of a particle's velocity and position.

Parameter ↓ \ Particle Index →	0	1	2	3	4
$x_{j,t}$	1	17	4	3	21
$v_{j,t}$	0	0	0	0	0
$p_{nb,t}$	1	2	5	17	21
$p_{l,t}$	1	13	5	8	21
$v_{l,j,t}$ =round(sqrt[$(p_{l,\,j,\,t} - x_{j,t})^2$])	0	4	1	5	0
$p_{g,t}$	1	2	12	13	21
$v_{g,j,t}$ = round(sqrt[$(p_{g,\,j,\,t} - x_{j,\,t})^2$])	0	15	8	10	0
$v_{t+1} = v_{t,j} + v_{l,j,t} + v_{g,j,t}$	0	19	9	15	0
$v_{t+1,Norm}$	0	0.44	0.209	0.349	0
$v_{t+1,Cum\text{-}Norm}$	0	0.44	0.651	1	1
rand = 0.79, select index<=$v_{t+1,CumNorm}$				X	
select value of $p_{nb,t}$ =17 at index position 3	1	2	5	17	21
swap 17 value in x_t(2) into x_t(3) by swapping it with 3 at x_t(3) Note: This operation reflects equation (7) in section 2.2	1	3	4	17	21

A summary of the ARPSO algorithm for the OP as previously described is:

Parameter settings and randomize initial swarm position of particles
Initial velocity for each particle equal to zero
Initial local, global, and neighborhood, best solution of particle exploration
Loop epoch ≤ MaxEpoch
Particle distance feasibility, shrinking, and expansion for "x_t" and "$p_{nb,t}$"
Evaluate objective function value of "x_t" and "$p_{nb,t}$"
Evaluate local, global, neighborhood solution of particles explored
if (epoch < MaxEpoch)
Attraction or repulsive direction for swarm
Compute velocity " $v_{lb,t}$" " $v_{gb,t}$," and "v_t"
Normalize velocity
Particle update ---swap element of p_{nb} into target particle
Neighborhood "$p_{nb,t}$" for each particle
Calculate swarm diversity

Table 2. Particle distance feasibility, shrinking and expansion.

Index	1	2	3	…	16	17	18	19	20	21
Tour	1	5	13	…	14	10	3	11	9	21
Assume another updated particle $x_{i,t}$ has the above tour from node 1 to 21 that is > 15, and that $T_{max} \leq 15$. Since the T_{max} constraint is violated, we start deleting nodes.										
Index	1	2	3	4	5	6	18	19	20	21
Tour	1	5	13	6	4	21	10	3	11	9
Assume "feasible tour" above achieved after deleting several nodes to yield $T_{max} \leq 15$							Deleted nodes			
Index	1	2	3	4	5	…	18	19	20	21
Tour	1	5	13	6	4	…	3	11	9	21
"Particle expansion." Objective function value or tour score calculated only from feasible nodes at index 1 through 5. Note: (1) Associated score for nodes 1 and 21 is always zero in the OP. (2) At the start of particle expansion; a marker is placed on the index with the last feasible node before node 21 is relocated. Node 21 is then moved to index position 21 in the feasible tour. Lastly, all previously deleted nodes are added back (array expanded) between the last feasible node and node 21 in the order these nodes were previously discarded.										

3. Experimental Set-Up and Results

The preliminary experimental observations on various parameter settings of the ARPSO demonstrated best results with a swarm size of 100 particles exploring and socializing for 700 epochs with a neighborhood size of 50 per particle. The ARPSO is tested using problem set II with 21 nodes and 11 different T_{max} instances in Chao et al. [3]. The simulation experiments of the PSO were implemented using the MatLab script software on an AMD Athlon 64 processor running at 3.7 GHz with 512 MB of RAM.

To allow comparison of the ARPSO results obtained in this research with the results from Chao et al. [2] and our previous research [16], each one of the eleven problem instances were conducted with ten replications. The results are summarized in Table 3 and Figure 1.

Table 3. Preliminary experimental results on 21-node problem.

Problem Instance: T_{max}	ARPSO		PSO		Chao et al. [2]
	Best Score (Max)	Average (Max)	Best Score (Max)	Average (Max)	Best Score (Max)
15	120	117	110	113	120
20	200	199	130	119.5	200
23	200	200	155	139.5	210
25	210	198.5	160	146	230
27	230	230	185	156.5	230
30	230	230	175	168.5	265
32	260	238.5	205	179.5	300
35	295	277	205	195.5	320
38	340	325	255	225	360
40	365	353	240	225.5	395
45	440	417	275	251.5	450

Table 3 and Figure 1 demonstrate that ARPSO has significantly improved the PSO capability to solving the orienteering problem. The results illustrate ARPSO is able to obtain optimal or near optimal results compared to PSO. Moreover, with respect to best known published results, ARPSO is shown to be more competitive than PSO.

4. Conclusions

A modified PSO algorithm known as the ARPSO algorithm was implemented on a combinatorial benchmark problem, namely the OP, from Chao et al. [3]. In particular, OP set II having 21 nodes with various distance constraints (T_{max}) were used to exhibit the substantial improvement of ARPSO on PSO results obtained in Dallard et al. [16] in solving the OP. The ARPSO results are a momentous improvement compared to earlier results found by PSO [16], and in some cases, it finds the best known published solutions. The computational effort using ARPSO for T_{max} values 15 to 45 varied from <1 minute to 30 minutes. In conclusion, the ARPSO was proven to be worthwhile in solving the OP and further parameter tuning will likely yield all of the best known published results for problem set II. The parameter modifications would also include analysis of an effective neighborhood size that maybe smaller than 50. Lastly, research on scalability issues for OP using ARPSO can be addressed by examining its results on the remaining problem sets consisting of 32 and 33 nodes and comparing ARPSO overall performance to other techniques yielding the best published results.

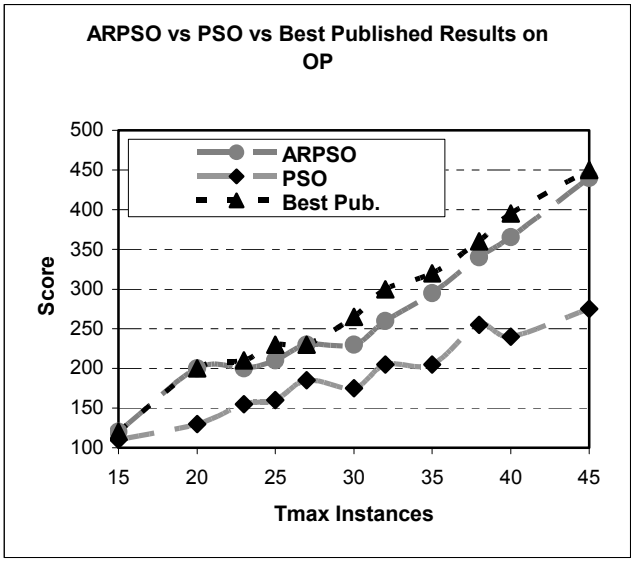

Figure 1. Performance comparison of ARPSO to PSO and best published results.

5. References

[1] B.C.H. Chang, A. Ratnaweera, S.K. Halgamuge, and H.C. Watson, "Particle Swarm Optimization for Protein Motif Discovery", *Genetic Programming and Evolvable Machines*, Springer Netherlands, 11/3/2004, pp. 203-214.

[2] I.M. Chao, B.L. Golden, and E.A. Wasil, "A Fast and Effective Heuristic for the Orienteering Problem", *European Journal of Operational Research*, Elsevier, Amsterdam, 1996, pp. 475-489.

[3] I.M. Chao, B.L. Golden, and E.A. Wasil, "The Orienteering Problem", *European Journal of Operational Research*, Elsevier, Amsterdam, 1996, pp. 464-474.

[4] M. Clerc, "Discrete Particle Swarm Optimization,", *New Optimization Techniques in Engineering*, Springer-Verlag, 2004.

[5] B.L. Golden, L. Levy, and R. Vohra, "The Orienteering Problem", *Naval Research Logistics*, 1987, pp. 307-318.

[6] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by Simulated Annealing", *Science*, 1983, pp. 671-680.

[7] C.P. Keller, "Algorithms to Solve the Orienteering Problem: A Comparison", *European Journal of Operational Research*, 1989, pp. 224-231.

[8] J. Kennedy, and R.C. Eberhart, "Particle Swarm Optimization," *Proceedings of the IEEE International Conference on Neural Networks*, IEEE Service Center, Piscataway, NJ, 1995, pp. 1942-1948.

[9] J. Kennedy, and R.C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, 2001.

[10] S. Kulturel-Konak, B.A. Norman, D.W. Coit, and A.E. Smith, "Exploiting Tabu Search Memory in Constrained Problems", *INFORMS Journal on Computing*, 2004, pp. 241-254.

[11] Y.C. Liang, S. Kulturel-Konak, and A.E. Smith, "Meta-Heuristics for the Orienteering Problem", *Proceedings of the Congress on Evolutionary Computation*, Honolulu, Hawaii, May 2002, pp. 384-389.

[12] M.F. Tasgetiren, and Y.C. Liang, "A Binary Particle Swarm Optimization Algorithm for Lot Sizing Problem", *Journal of Economic and Social Research*, 2003, pp. 1-20.

[13] M.F. Tasgetiren, Y.C. Liang, M. Sevkli, and G. Gencyilmaz, "Particle Swarm Optimization Algorithm for Makespan and Maximum Lateness Minimization in Permutation Flowshop Sequencing Problem", *4th International Symposium on Intelligent Manufacturing Systems*, Sakarya, Turkey, September 2004, pp. 431-441.

[14] M.F. Tasgetiren, Y.C. Liang, M. Sevkli, and G. Gencyilmaz, "Particle Swarm Optimization Algorithm for Makespan and Total Flow time Minimization in Permutation Flow shop Sequencing Problem", *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2004, pp. 382-389.

[15] G. Venter, and J. Sobieszczanski-Sobieski, "Particle Swarm Optimization", *American Institute of Aeronautics and Astronautics*, 2002, pp. 1202-1235.

[16] H. Dallard, S. Lam, and S. Kulturel-Konak, "A Particle Swarm Optimization Approach to the Orienteering Problem", *Proceedings of Industrial Engineering Research Conference (IERC)*, Orlando, FL, 2006.

[17] J. Riget, and J.S. Vesterstrom, "A Diversity Guided Particle Swarm Optimizer—the ARPSO", *EVA-Life Technical Report,* (http://www.evalife.dk), 2002.

[18] X. Hu, R.C. Eberhart, and Y. Shi, "Swarm Intelligence for Permutation Optimization: A Case Study of n-Queens Problem", *Proceedings of the IEEE Swarm Intelligence Symposium,* 2003, pp. 243-246.