International Journal of Industrial Engineering, 18(2), 92-102, 2011.

DISCRETE PARTICLE SWARM OPTIMIZATION FOR THE ORIENTEERING PROBLEM

Shanthi Muthuswamy[a], and Sarah Lam[b]

[a]Department of Technology, Northern Illinois University, DeKalb, Illinois 60115, U.S.A.
[b]Systems Science and Industrial Engineering Department, Binghamton University, Binghamton, New York 13902, U.S.A.
Corresponding author: Shanthi Muthuswamy, smuthuswamy@ niu.edu

Discrete particle swarm optimization (DPSO) is gaining popularity in the area of combinatorial optimization in the recent past due to its simplicity in coding and consistency in performance. A DPSO algorithm has been developed for orienteering problem (OP) which has been shown to have many practical applications. It uses reduced variable neighborhood search as a local search tool. The DPSO algorithm was compared with ten heuristic models from the literature using benchmark problems. The results show that the DPSO algorithm is a robust algorithm that can optimally solve the well known OP test problems.

Keywords: Discrete particle swarm optimization, reduced variable neighborhood search, orienteering problem.

(Received 1 Jul 2010; Accepted in revised form 6 Feb 2011)

1. INTRODUCTION

Particle Swarm Optimization (PSO) is an evolutionary optimization algorithm that is inspired by nature. PSO has been widely used to optimize several continuous nonlinear functions as well as combinatorial problems. Many researchers are using PSO as a tool for solving combinatorial problems due to its simplicity in structure, ease of implementation and performance robustness (Pan et al., 2008). Orienteering Problem (OP), which is also called the generalized traveling salesman problem, is a combinatorial optimization problem. OP has been proved to be a NP-hard problem (Golden et al., 1987). A limited number of exact methods have been developed to solve smaller problem instances of the OP. Many heuristic approaches have been implemented for larger sized OPs. Various adaptive optimization heuristics such as tabu search, genetic algorithm, and ant colony optimization algorithms (discussed in section 2) have been developed for the OP. In this paper a DPSO algorithm has been designed for OP. The performance of the DPSO heuristic was measured against other metaheuristic algorithms from the literature using benchmark problems.

1.1 Particle Swarm Optimization

Particle swarm optimization, inspired by the social behavior of bird flocking and fish schooling, is a population based metaheuristic introduced by Kennedy and Eberhart (1995). In PSO, each particle represents a solution and the swarm of particles flies through the search space in an effort to reach the global optimum. All members (particles) of the population are maintained throughout the search procedure and their information is socially shared among the individuals to direct the search toward the best position in the search space. The particles fly through the multi-dimensional problem space with specific velocities and follow the current best known particles. During flight each particle adjusts its position according to its own experience and the experience of the neighboring particles. The neighborhood could encompass either a local neighborhood or the global neighborhood which forms the two variation of the fundamental PSO algorithm. The particle's best solution is called the p_{ibest}, global best is called the g_{best} and the restricted neighborhood's best particle is named the l_{best} (Kennedy and Eberhart, 2001). In each iteration the particle flies to the next position with a certain velocity using the p_{ibest} and g_{best} values as shown in equations 1 and 2 (Shi and Eberhart, 1999). Replace g_{best} with l_{best} for the restricted neighborhood version of the PSO. PSO combines local search methods with global search techniques thus balancing between exploitation and exploration.

$$v_i^{k+1} = w.v_i^k + c_1 r_1 \left(p_{ibest} - x_i^k\right) + c_2 r_2 \left(g_{best} - x_i^k\right) \quad \ldots \quad (1)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad \ldots \quad (2)$$

where

x_i^k is the position of the i^{th} particle at iteration k, v_i^k is the velocity of the i^{th} particle at iteration k, w is the inertia parameter, c_1 is the cognitive parameter, c_2 is the social parameter, r_1 and r_2 are random numbers.

1.2 Discrete Particle Swarm Optimization

Discrete PSO (DPSO) is a modified version of PSO which applies discrete or qualitative distinction between variables. Kennedy and Eberhart (1997) developed the first DPSO with binary valued particles. Since then several versions of DPSO have been developed. DPSO will facilitate solving the combinatorial optimization problems due to its ease of implementation, simple structure and its robustness (Pan et al., 2008; Tasgetiren, 2007). DPSO algorithms that have been implemented so far can be broadly classified in to five categories namely (i) binary valued DPSO (Kennedy and Eberhart, 1997; Liao et al., 2007; Pan et al., 2008), (ii) DPSO with dummy variable to transition from combinatorial to discrete state and vice-versa (Jarboui et al., 2007; Jarboui et al., 2008a; Jarboui et al., 2008b), (iii) DPSO with crossover and mutation techniques (Lian et al., 2006; Lian et al., 2008; Pan et al., 2008), (iv) modified continuous PSO with smallest position value rule (Chen et al., 2006; Tasgetiren et al., 2007) and, (v) the miscellaneous DPSO models which includes the other versions of DPSO algorithm seen in the literature such as quantum particle version DPSO (Pang et al., 2004), fuzzy DPSO (Anghinolfi and Paolucci, 2009), DPSO with pseudo-insertion and extract-reinsert operators (Hu et al., 2004) to name a few.

1.3 Industry Applications of PSO Algorithm

Since 2002, research applying PSO has grown rapidly. The number of papers using PSO as an optimization tool has totaled over 300 until 2004 (Venter and Sobieski, 2002) and is growing exponentially ever since. Popularity of PSO is due to its several strengths namely (i) very few parameters to adjust, (ii) simple structure, (iii) ease of implementation, (iv) robustness, and (v) convergence speed. Some of the PSO applications include communication satellite design (Abido, 2002), power and voltage control problem (Fukuyama et al., 1999; Yeh, 2003), supplier selection and ordering problem (Van den Bergh and Engelbecht 2000), neural network training (Brandstatter and Baumgartner, 2002), mass spring system (Allahverdi and Al-Anzi, 2006), Golinski speed reducer problem, and Rosenbrock function problem (Abido, 2002). Combinatorial optimization problems that have been solved using DPSO include vehicle routing problem (Pang et al., 2004), traveling salesman problem (Venter and Sobieski, 2002), scheduling problem (Anghinolfi and Paolucci, 2009; Jarboui et al., 2008a; Jarboui et al., 2008b; Lian et al., 2006; Lian et al., 2008; Pan et al., 2008; Tasgetiren et al., 2007; Tseng and Liao, 2008; Jin et al., 2007), clustering problem (Jarboui et al., 2007), transmission network expansion problem (Jin et al., 2007), and orienteering problem (Dallard et al., 2006; Dallard et al., 2007; Sevkli et al., 2007).

This paper has been structured into the following sections: section 2 describes the problem under study (OP); section 3 gives a detailed description of the DPSO algorithm; section 4 expands upon the parameter value selection process; section 5 discusses the results of the DPSO algorithm performance and section 6 provides the concluding remarks along with the future extension recommendations.

2. PROBLEM DESCRIPTION

The orienteering problem initiated from the sport of orienteering which involves cross-country running and navigation through a forest using a map and compass. There are several control points or nodes to be visited and each node has a score tied to it. Lower scores are usually allocated to nodes near the start and finish areas and larger scores to those further away. The difficulty of reach and the distance of the node with respect of other nodes are also taken into consideration while allocating scores to control points. The competitors start and end their tour at specified control points (which do not have any scores). The term OP which was introduced by Tsiligirides (1984) is the *score orienteering event* version of the sport in which the competitors do not have to visit all the nodes. The objective is to maximize the score within a certain prescribed time limit. Competitors who arrive at the terminal point after the prescribed time limit are disqualified.

2.1 Industry Applications of Orienteering Problem

Literature reveals several applications of OP including routing oil tankers to service ships (Golden et al., 1987), customer vehicle assignment problem (Golden et al., 1981; Golden et al., 1984), inventory routing problem (Golden et al., 1981; Golden et al., 1984), production scheduling (Balas, 1989), bank/postal delivery and industrial refuse collection problems (Kantor and Rosenwein, 1992), and single ring design problem while constructing telecommunication networks (Thomadsen and Stidsen, 2003). Recent applications of OP include mobile tourist guide application (Souffriau et al., 2008; Wang et al., 2008) to help tourists pick the most valuable attractions to visit within the given time span of their visit, and a military application (Wang et al., 2008) to aid surveillance aircrafts to choose a subset of places to photograph within the given amount of time or fuel constraint.

Since OP is a NP hard problem a few exact methods using branch and bound (Ramesh et al., 1992; Ramesh et al., 1992b), branch and cut (Fischetti et al., 1998), Lagrangean relaxation (Balas, 1989), and minimum directed 1-subtree relaxation (Kataoka et al., 1998) have been developed in the past. Several heuristic approaches have been developed in the last two decades. Golden et al. (1987) implemented a heuristic with center of gravity technique. Golden et al. (1988) improved the center of gravity technique with learning capabilities. Keller (1989) altered his algorithm using Multi-Objective Vending Problem (MVP) to solve the OP. Ramesh and Brown (1992) created a four phase heuristic to solve the OP. Chao et al. (1996) developed a 'fast and effective heuristic' for OP. Many metaheuristic algorithms such as Genetic Algorithm (GA) (Tasgetiren, 2002), Ant Colony Optimization (ACO) (Ke et al., 2008; Liang et al., 2002; Liang and Smith, 2006), Tabu Search (TS) (Kulturel-Konak et al., 2004; Liang et al., 2002), and PSO (Dallard et al., 2006; Dallard et al., 2007; Sevki et al., 2007) have also been implemented to solve the OP. In this paper a novel DPSO algorithm has been introduced which calculates the new position of the particle using p_{ibest}, g_{best} and the current position of the particle using a discrete representation as discussed in the following section.

3. DPSO ALGORITHM

In the DPSO algorithm of OP each particle constitutes a tour encompassing the list of nodes visited such that T_{max}, the distance constraint was obeyed. The starting and the ending nodes are distinct and specified. In order to ensure a good starting solution in the population, the first particle was built using the *s/d* (score/distance) ratio. Starting from the first node, the feasible node with the highest *s/d* value was chosen as the next city to be visited. Guided by the *s/d* values a feasible tour was constructed for the first particle. The initial solutions for the remaining particles were constructed randomly.

The new position (new tour) for each particle was calculated as follows: For each node in the current particle a random number Λ is generated. If $\Lambda < w$, the node is accepted as a part of a temporary array, Ω. In a similar fashion nodes were included in the temporary array from the p_{ibest} particle if $\Lambda < c_1$, and the g_{best} particle if $\Lambda < c_2$. Duplicate nodes were removed from Ω. If the temporary array was feasible it was accepted as the new position of the particle otherwise nodes were randomly deleted till a feasible solution was reached. In order to improve the efficacy of the algorithm so as to prevent it from being caught in local optima a Reduced Variable Neighborhood Search (RVNS) was used as a local search tool (Sevkli and Sevilgen, 2006). In RVNS the solution space is searched by switching neighborhoods. The two neighborhoods used in the algorithm include *insert* and *exchange*. In the insert neighborhood a new node is inserted in the existing tour while in the exchange neighborhood a new node is exchanged for an existing node in the tour. After the completion of the RVNS procedure a 2-opt operation is performed to optimize the tour. If the tour length has been reduced by the 2-opt method, a node insert procedure is conducted in an attempt to increase the fitness value (total score) of the tour. The flowchart for the DPSO algorithm is given in Figure 1 and the pseudo code for RVNS is shown in Figure 2 (Sevkli et al., 2007).

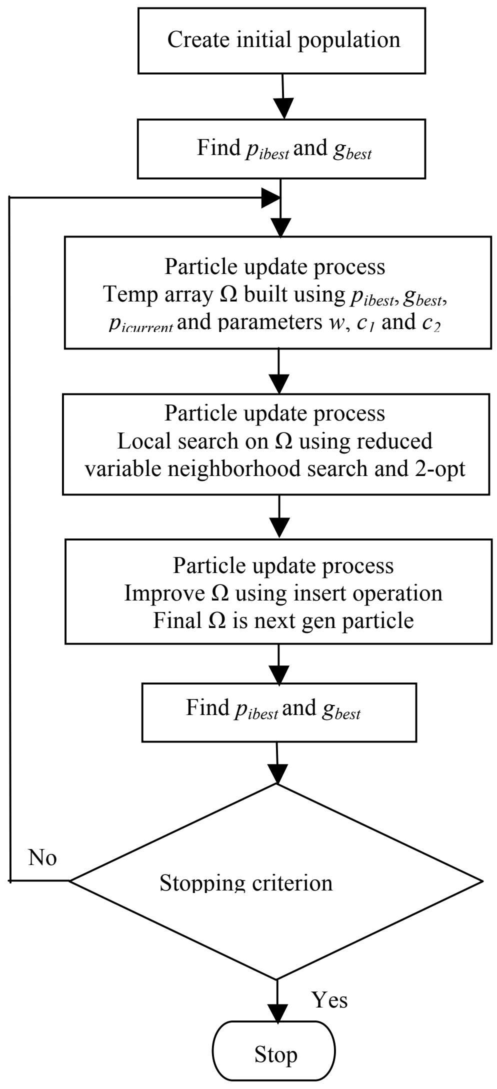

Create initial population
Find p_{ibest} and g_{best}
Particle update process
Temp array Ω built using p_{ibest}, g_{best}, $p_{icurrent}$ and parameters w, c_1 and c_2
Particle update process
Local search on Ω using reduced variable neighborhood search and 2-opt
Particle update process
Improve Ω using insert operation
Final Ω is next gen particle
Find p_{ibest} and g_{best}
No
Stopping criterion
Yes
Stop

Figure 1. DPSO algorithm flowchart

```
Initial solution, $S_{init}$
New solution, $S_{new}$
Initial solution tour length, $S_{init\_len}$
New solution tour length, $S_{new\_len}$
**while** stopping criterion not satisfied
    neighborhood ←1
    **if** neighborhood =1
        $S_{new}$ ←Insert ($S_{init}$)
    **end if**
    **if** neighborhood =2
        $S_{new}$ ← Exchange ($S_{init}$)
    **end if**
    **if** Fitness($S_{new}$) > Fitness($S_{init}$) | $S_{new\_len} < S_{init\_len}$
        $S_{init} \leftarrow S_{new}$
        $S_{init\_len} \leftarrow S_{new\_len}$
        neighborhood ←1
    **else**
        neighborhood ←2
    **end if**
**end while**
```

Figure 2. RVNS pseudo code

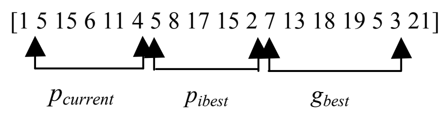
In order to better comprehend the DPSO algorithm the construction of the temporary array Ω is illustrated in Figure 3(a). Based on $\Lambda(x_i^n)$, $\Lambda(x_p^n)$ and $\Lambda(x_g^n)$ values assume Ω to be Figure 3(b). Remove duplicates to get the updated value of Ω as shown in Figure 3(c). Using the updated Ω the next position of the particle is constructed as shown in the flowchart (Figure 1).

1	5	8	15	6	3	20	19	11	7	4	21	$p_{current}$

1	5	8	14	9	3	20	17	15	2	21	p_{ibest}

1	7	13	15	9	8	18	19	5	3	4	2	12	21	g_{best}

(a)


[1 5 15 6 11 4 5 8 17 15 2 7 13 18 19 5 3 21]

(b)

[1 6 11 4 8 17 15 2 7 13 18 19 5 3 21]

(c)

Figure 3. DPSO particle (a) DPSO particle configuration (b) Temporary array Ω (c) Updated Ω

4. PARAMETER VALUE SELECTION

For the DPSO algorithm of OP, six parameters including the population size, number of generations, w, c_1, c_2, and the stopping criterion were chosen using Design of Experiments (DOE). The stopping criterion was the number of generations for which the best solution found remains unchanged. Based on preliminary experimentation two levels were chosen for the six factors as shown in Table 1.

Table 1. DOE Parameters and Levels

Factor #	Factors	Level 1	Level 2
1	Number of particles	30	40
2	Number of generations	50	100
3	w	0.4	0.8
4	c1	0.4	0.8
5	c2	0.9	1
6	stopping criterion	25	40

The experimentation was carried out using a 2^6 full factorial design. Each factor level combination was tested for 10 replications. The response variables used were the Relative Percentage Error (RPE) and the Average Relative Percentage Error (ARPE). RPE is defined as the error between the best known solution and the best solution of all ten replications. It shows if the algorithm can find the best known solution.

$$RPE = \frac{best\ known\ \ score - best\ score}{best\ known\ \ score} * 100 \quad \ldots \quad (3)$$

ARPE is defined as the average error of all replications. It shows the robustness of the algorithm.

$$ARPE = \frac{\sum_{i=1}^{10}\left(\frac{best\ known\ score - i^{th} replication\ score}{best\ known\ score} * 100\right)}{10} \quad \ldots \quad (4)$$

Figure 4 shows the main effects plots of the Analysis of Variance (ANOVA) results.

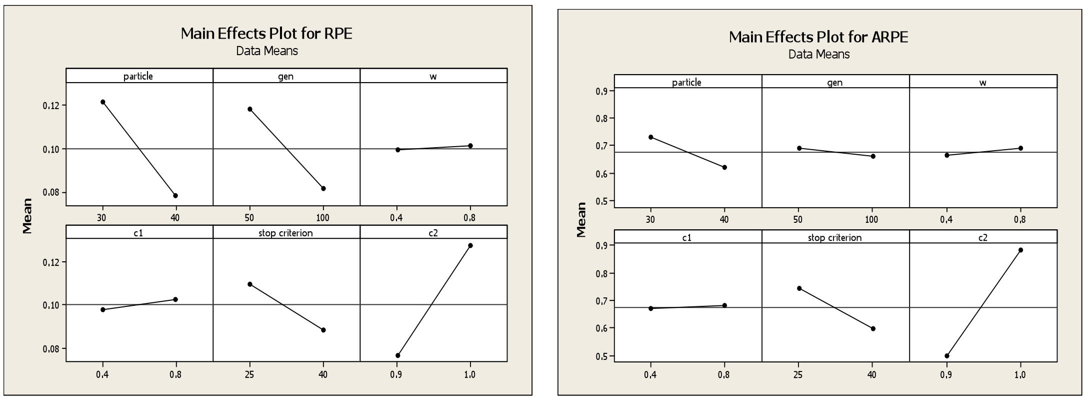


Figure 4. ANOVA main effects plots (a) For response variable RPE (b) For response variable ARPE

Figure 4(a) shows that the parameters, number of particles, number of generations, c_2, and the stopping criterion significantly affect the response variable RPE. Similarly Figure 4(b) shows that the parameters, number of particle, c_2, and the stopping criterion influence the ARPE more than the other parameters. The parameter values of the DPSO (see Table 2) were chosen such that the overall RPE and ARPE were minimized.

Table 2. DPSO Parameters and their Values

Factor #	Factors	Value
1	Number of particles	40
2	Number of generations	100
3	w	0.4
4	c1	0.4
5	c2	0.9
6	stopping criterion	40

5. RESULTS AND DISCUSSION

The performance of the DPSO algorithm was validated using four benchmark problem sets from the literature totaling to 67 problems. Tsiligirides (1984) developed the three problem sets namely, dataset 1 with 32 nodes (18 problems), dataset 2 with 21 nodes (11 problems), and dataset 3 with 33 nodes (20 problems). Chao et al. (1996) corrected an error in Tsiligirides's dataset 1 and named it dataset 4 (32 nodes with 18 problems). The size of the entire search space (including feasible solutions, and infeasible solutions that violate the T_{max} constraint) for these problems were 3.3 x 10^{17} (21 node problem), 7.2 x 10^{32} (32 node problem), and 2.2 x 10^{34} (33 node problem). The results of the DPSO algorithm were compared with ten different heuristic models from the literature. Table 3 summarizes the acronym abbreviations of various heuristic models that were used for comparison. The experiments were conducted on a windows XP environment on an Intel CPU T2400@1.83 GHz processor using 1 GB RAM. The code was implemented using MATLAB 7.1.0.

Table 3. Acronym Abbreviations for the Model Names

Acronym abbreviations	
Tmax	Maximum tour length allowed
UB	Upper bound on score (Leifer and Rosenwein, 1994)
TS	Tsiligirides's heuristic (Tsiligirides, 1984)
TC	Tsiligirides's heuristic implemented by Chao et al. (Chao et al., 1996)
MVP	Keller's MVP heuristic (Keller, 1989)
GLV	Golden et al.'s heuristic (Golden et al., 1987)
GWL	Golden et al.'s heuristic (Golden et al., 1988)
ANN	Wang et al.'s artificial neural network model (Wang et al., 1995)
CGW	Chao et al.'s heurisitc (Chao et al., 1996)
GA	Tasgetiren's genetic algorithm (Tasgetiren, 2002)
ACO	Liang and Smith's ant colony optimization model (Liang and Smith, 2006)
Tabu	Kulturel-Konak et al.'s tabu model (Kulturel-Konak et al., 2004)
PSO	Sevkli et al.'s PSO algorithm (Sevkli et al., 2007)
DPSO	New DPSO algorithm proposed

Tables 4, 5, 6 and 7 provide the comparison of the best solution of 10 replications obtained by the DPSO algorithm against the best solution of various heuristic models for problem sets 1 through 4. A '+' symbol denotes that the DPSO algorithm found a superior solution than the heuristic model in comparison and vice-versa for the '-' sign. The results in the tables show that the DPSO algorithm is competitive. It was able to reach the best known solutions for all of the 67 problem instances.

Table 4 . Comparison of DPSO Algorithm with the Other Heuristic Models for Problem Set 1

		Compared with heuristics from literature										DPSO vs. other heuristics								
Tmax	UB	TC	MVP	GLV	GWL	ANN	CGW	GA	ACO	Tabu	DPSO	TC	MVP	GLV	GWL	ANN	CGW	GA	ACO	Tabu
5	10	10	10	10	10	10	10	10	10	10	10									
10	20	15	15	15	15	15	15	15	15	15	15									
15	45	45	45	45	45	45	45	45	45	45	45									
20	70	65	65	65	65	65	65	65	65	65	65									
25	95	90	90	90	90	90	90	90	90	90	90									
30	120	110	110	110	110	110	110	110	110	110	110									
35	140	135	130	125	135	135	135	135	135	135	135		+	+						
40	160	150	155	140	155	155	155	155	155	155	155	+		+						
46	180	170	175	165	175	175	175	175	175	175	175	+		+						
50	195	185	185	180	190	190	190	190	190	190	190	+	+	+						
55	210	195	200	200	205	205	205	205	205	205	205	+	+	+						
60	230	220	225	205	225	225	225	225	225	225	225	+		+						
65	245	235	240	220	240	240	240	240	240	240	240	+		+						
70	260	255	260	240	260	260	260	260	260	260	260	+		+						
73	270	260	265	255	265	265	265	265	265	265	265	+		+						
75	270	265	270	260	270	270	270	270	270	270	270	+		+						
80	285	270	280	275	280	280	280	280	280	280	280	+		+						
85	285	280	285	285	285	285	285	285	285	285	285	+								
Summary of DPSO model vs. other heursitics											+	11	3	11	0	0	0	0	0	0
											-	0	0	0	0	0	0	0	0	0

Table 5. Comparison of DPSO Algorithm with the Other Heuristic Models for Problem Set 2

		Compared with heuristics from literature										DPSO vs. other heuristics								
Tmax	UB	TS	MVP	GLV	GWL	ANN	CGW	GA	ACO	Tabu	DPSO	TS	MVP	GLV	GWL	ANN	CGW	GA	ACO	Tabu
15	145	120	120	120	120	120	120	120	120	120	120									
20	200	190	200	200	200	200	200	200	200	200	200	+								
23	215	205	210	210	205	205	210	210	210	210	210	+			+	+				
25	240	230	230	230	230	230	230	230	230	230	230									
27	265	230	230	230	230	230	230	230	230	230	230									
30	275	250	260	260	265	265	265	265	265	265	265	+	+	+						
32	305	275	300	260	300	300	300	300	300	300	300	+		+						
35	350	315	320	300	320	320	320	320	320	320	320	+		+						
38	375	355	360	355	360	360	360	360	360	360	360	+		+						
40	400	395	380	380	395	395	395	395	395	395	395		+	+						
45	450	430	450	450	450	450	450	450	450	450	450	+								
Summary of DPSO model vs. other heursitics											+	7	2	5	1	1	0	0	0	0
											-	0	0	0	0	0	0	0	0	0

Table 6. Comparison of DPSO Algorithm with the Other Heuristic Models for Problem Set 3

		Compared with heuristics from literature										DPSO vs. other heuristics								
Tmax	UB	TS	MVP	GLV	GWL	ANN	CGW	GA	ACO	Tabu	DPSO	TS	MVP	GLV	GWL	ANN	CGW	GA	ACO	Tabu
15	175	100	170	170	170	170	170	170	170	170	170	+								
20	210	140	200	200	200	200	200	200	200	200	200	+								
25	290	190	260	250	260	250	260	260	260	260	260	+		+						
30	340	240	320	320	320	320	320	320	320	320	320	+								
35	395	290	370	380	390	390	390	390	390	390	390	+	+	+						
40	445	330	430	420	430	420	430	430	430	430	430	+		+						
45	490	370	460	450	470	470	470	470	470	470	470	+	+	+						
50	535	410	520	500	520	520	520	520	520	520	520	+		+						
55	575	450	550	520	550	550	550	550	550	550	550	+		+						
60	605	500	570	580	580	580	580	580	580	580	580	+	+							
65	635	530	610	600	610	610	610	610	610	610	610	+		+						
70	665	560	640	640	640	640	640	640	640	640	640	+								
75	695	590	670	650	670	670	670	670	670	670	670	+		+						
80	725	640	700	690	710	700	710	710	710	710	710	+	+	+						
85	750	670	740	720	740	740	740	740	740	740	740	+		+						
90	785	690	760	770	770	770	770	770	770	770	770	+	+							
95	800	720	790	790	790	790	790	790	790	790	790	+								
100	800	760	800	800	800	800	800	800	800	800	800	+								
105	800	770	800	800	800	800	800	800	800	800	800	+								
110	800	790	800	800	800	800	800	800	800	800	800	+								
Summary of DPSO model vs. other heursitics											+	20	5	10	0	3	0	0	0	0
											-	0	0	0	0	0	0	0	0	0

Table 7. Comparison of DPSO Algorithm with the Other Heuristic Models for Problem Set 4

	Compared with heuristics from literature								DPSO vs. other heuristics						
Tmax	TS	TC	ANN	CGW	GA	ACO	Tabu	DPSO	TS	TC	ANN	CGW	GA	ACO	Tabu
5	10	10	10	10	10	10	10	10							
10	15	15	15	15	15	15	15	15							
15	45	45	45	45	45	45	45	45							
20	65	65	65	65	65	65	65	65							
25	90	85	90	90	90	90	90	90		+					
30	110	110	110	110	110	110	110	110							
35	135	135	130	135	135	135	135	135			+				
40	150	150	155	155	155	155	155	155	+	+					
46	175	175	175	175	175	175	175	175							
50	190	185	190	190	190	190	190	190		+					
55	205	200	205	205	205	205	205	205		+					
60	220	220	220	220	225	225	225	225	+	+	+	+			
65	240	240	240	240	240	240	240	240							
70	255	250	260	260	260	260	260	260	+	+					
73	260	265	265	265	265	265	265	265	+						
75	270	265	270	275	270	275	275	275	+	+	+		+		
80	275	270	280	280	280	280	280	280	+	+					
85	280	285	285	285	285	285	285	285	+						
Summary of DPSO model vs. other heursitics								+	7	8	3	1	1	0	0
								-	0	0	0	0	0	0	0

Table 8 summarizes the total number of problems in which the DPSO algorithm outperforms the other heuristic models. A '-' symbol indicates the heuristic is not applicable to that dataset. The DPSO algorithm was at par with the ACO (Liang and Smith, 2006) and TS (Kulturel-Konak et al., 2004) models for all the datasets. It outperformed the GA (Tasgetiren, 2002), Tsiligirides's heuristic implemented by Chao et al. (1996), and the Golden et al.'s heuristic (Golden et al., 1988) in one instance each. In comparison to the other models the DPSO performed far better as shown in Table 8.

Table 8. Summary of Problems in Which DPSO Outperformed Other Heuristics

	Total number of problems in which DPSO outperformed other heuristics			
	Problem set 1	Problem set 2	Problem set 3	Problem set 4
TS	-	7	20	7
TC	11	-	-	8
MVP	3	2	5	-
GLV	11	5	10	-
GWL	0	1	0	-
ANN	0	1	3	3
CGW	0	0	0	1
GA	0	0	0	1
ACO	0	0	0	0
Tabu	0	0	0	0

Sevkli et al.'s PSO model (2007) contains two error metrics RPE and ARPE to measure the robustness of their model. The RPE and the ARPE were calculated for the 67 problems using the DPSO algorithm and compared with Sevkli et al.'s PSO model. Table 9 shows that both the models reach the best known solutions with their RPE values of zero. However, the ARPE of the DPSO algorithm is 0.84% lower than the PSO algorithm indicating that the DPSO algorithm is more robust than the PSO model.

Table 9. DPSO vs. PSO Model Comparison

	PSO	**DPSO**
RPE	0	0
ARPE	1.05	0.21

On an average the DPSO algorithm took 21.66 seconds to solve an OP instance. The CPU time comparison between different models has not been done since it is a machine and language/tool dependent criterion.

6. CONCLUSIONS

In this paper a DPSO algorithm was presented for the OP. In order to enhance the efficacy of the algorithm RVNS technique was used as a local search tool and 2-opt was performed to further optimize the solution. This algorithm was compared with ten different heuristic models from the literature using the benchmark datasets. The DPSO algorithm was able to find the best known solutions for all the problems and outperformed seven of the heuristics in one or more problem instances. In comparison to the PSO model from the literature the DPSO algorithm's ARPE was 0.84% lower which exhibits the robustness of the DPSO algorithm. As a future extension this DPSO algorithm could be enhanced to represent the team orienteering problem and other routing problems.

7. REFERENCES

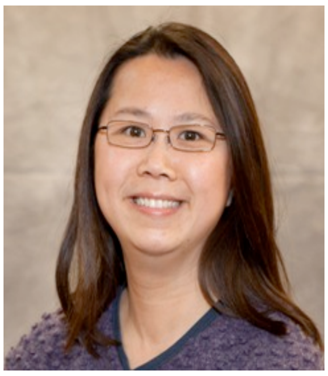
1. Abido, M. A. (2002). Optimal power flow using particle swarm optimization. Electrical Power Energy Systems, 24:563–571.
2. Allahverdi, A., Al-Anzi, F. S. (2006). A PSO and tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application. Computers & Operations Research, 33:1056–1080.
3. Anghinolfi, D., Paolucci, M. (2009). A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times. European Journal of Operational Research, 193(1):73–85.
4. Balas, E. (1989).The prize collecting traveling salesman problem. Networks, 19, 621–636.
5. Brandstatter, B., Baumgartner, U. (2002). Particle swarm optimization – mass-spring system analogon. IEEE Transactions on Magnetics, 38:997–1000.

6. Chao, I-M., Golden, B. L., & Wasil, E. A. (1996). A fast and effective heuristic for the orienteering problem. European Journal of Operational Research, 88:475–489.
7. Chen, A., Yang, G., & Wu, Z. (2006). Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. Journal of Zhejiang University Science A, 7(4):607–614.
8. Dallard, H., Lam, S., & Kulturel-Konak, S. (2006). A particle swarm optimization approach to the orienteering problem. Proceedings of Industrial Engineering Research Conference, Orlando, FL.
9. Dallard, H., Lam, S., & Kulturel-Konak, S. (2007). Solving the orienteering problem using attractive and repulsive particle swarm optimization. Proceedings of the International Conference on Information Reuse and Integration, Las Vegas, NV.
10. Fischetti, M., Gonzalez, J. J. S., & Toth, P. (1998). Solving the orienteering problem through branch-and-cut. INFORMS Journal of Computing, 10(2):133–148.
11. Fukuyama, Y., Takayama, S., Nakanishi, Y., & Yoshida, H. (1999). A particle swarm optimization for reactive power and voltage control in electric power systems. Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, FL.
12. Golden, B. L., Assad, A., & Dahl, R. (1984). Analysis of a large-scale vehicle routing problem with an inventory component. Large Scale Systems, 7:181–190.
13. Golden, B. L., Levy, L., & Dahl, R. (1981). Two generalizations of the traveling salesman problem. Omega, 8(4):439–441.
14. Golden, B. L., Levy, L., & Vohra, R. (1987).The orienteering problem. Naval Research Logistics, 34,307–318.
15. Golden, B. L., Wang, Q., & Liu, L. (1988). A multifaceted heuristic for the orienteering problem. Naval Research Logistics, 354:359–366.
16. Hu, X., Shi, Y., & Eberhart, R.C. (2004). Recent advances in particle swarm. Proceedings of IEEE Congress on Evolutionary Computation, 1: 90–97.
17. Jarboui, B., Cheikh, M., Siarry, P., & Rebai, A. (2007). Combinatorial particle swarm optimization (CPSO) for partitional clustering problem. Applied Mathematics and Computation, 192:337–345.
18. Jarboui, B., Damak, N., Siarry, P., & Rebai, A. (2008a). A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. Applied Mathematics and Computation, 195:299–308.
19. Jarboui, B., Ibrahim, S., Siarry, P., & Rebai, A. (2008b). A combinatorial particle swarm optimization for solving permutation flowshop problems. Computers & Industrial Engineering, 54:526–538.
20. Jin, Y-X., Cheng, H-Z.,Yan, J-Y., & Zhang, L. (2007). New discrete method for particle swarm optimization and its application in transmission network expansion planning. Electric Power Systems Research, 77:227–233.
21. Kantor, M., Rosenwein, M. (1992). The orienteering problem with time windows. Journal of the Operational Research Society, 43(6):629–635.
22. Kataoka, A., Yamda, T., & Morita, S. (1998). Minimum directed 1-subtree relaxation for score orienteering problem. European Journal of Operational Research, 104:139–153.
23. Ke, L., Archetti, C., & Feng, Z. (2008). Ants can solve the team orienteering problem. Computers and Industrial Engineering, 54(3):648–665.
24. Keller, C. P. (1989). Algorithms to solve the orienteering problem: A comparison. European Journal of Operational Research, 41:224–231.
25. Kennedy, J., Eberhart, R. C. (1995). Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, 1942–1948.
26. Kennedy, J., Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, 4104–4109.
27. Kennedy, J., Eberhart, R. C., & Shi, Y. (2001). Swarm intelligence. Morgan Kaufmann: San Mateo.
28. Kulturel-Konak, S., Norman, B. A., Coit, D. W., & Smith, A. E. (2004). Exploiting tabu search memory in constrained problems. INFORMS Journal of Computing, 16(3):241–254.
29. Leifer, A. C., Rosenwein, M. S. (1994). Strong linear programming relaxations for the orienteering problem. European Journal of Operational Research, 73:517–523.
30. Lian, Z., Gu, X., & Jia, B. (2008). A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan. Chaos Solitons & Fractals, 35:851–861.
31. Lian, A., Jiao, B., & Gu, X. (2006). A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan. Applied Mathematics and Computation, 183:1008–1017.
32. Liang, Y-C., Kulturel-Konak, S., & Smith, A. E. (2002). Meta heuristics for the orienteering problem. Proceedings of the 2002 Congress on Evolutionary Computation, 384–389.
33. Liang, Y-C., Smith, A. E. (2006). An ant colony approach to the orienteering problem. Journal of the Chinese Institute of Industrial Engineers, 23:403–414.
34. Liao, C-J., Tseng, C-T., & Luarn, P. (2007). A discrete version of particle swarm optimization for flowshop scheduling problems. Computers & Operations Research, 34:3099–3111.

35. Pan, Q-K., Tasgetiren, M. F., & Liang, Y-C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Computers & Operations Research, 35:2807–2839.
36. Pang, W., Wang, K-P., Zhou, C-G., & Dong, L-J. (2004). Fuzzy discrete particle swarm optimization for solving traveling salesman problem. Proceedings of the Fourth International Conference on Computer and Information Technology, 796–800.
37. Ramesh, R., Brown, K. M. (1992). An efficient four-phase heuristic for the generalized orienteering problem. Computers & Operations Research, 18(2):151–165.
38. Ramesh, R., Yoon, Y. S., & Karwan, M. H. (1992). An optimal algorithm for the orienteering tour problem. ORSA Journal on Computing, 4(2):155–165.
39. Sevkli, Z., Sevilgen, F. E. (2006).Variable neighborhood search for the orienteering problem. Proceedings of International Symposium on Computer and Information Sciences, Istanbul, Turkey.
40. Sevkli, Z., Sevilgen, F. E.,& Keles, O. (2007). Particle swarm optimization for the orienteering problem. International Symposium on Innovations in Intelligent Systems and Applications, Istanbul, Turkey.
41. Shi, Y., Eberhart, R. C. (1999). Empirical study of particle swarm optimization. Proceedings of Congress of Evolutionary Computation, 1945–1950.
42. Souffriau, W., Vansteenwegen, P., Vertommen, J., Vanden Berghe, G., & Van Oudheusden, D. (2008). A personalised tourist trip design algorithm for mobiletourist guides. Applied Artificial Intelligence, 22(10):964–985.
43. Tasgetiren, M. F. (2002). A genetic Algorithm with an adaptive penalty function for the orienteering problem. Journal of Economic and Social Research, 4(2):1–26.
44. Tasgetiren, M. F., Liang, Y-C., Sevkli, M., & Gencyilmaz, G. (2007). A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. European Journal of Operational Research, 177:930–947.
45. Thomadsen, T., Stidsen, T. (2003). The quadratic selective travelling salesman problem. Informatics and Mathematical Modelling Technical Report 2003-17, Technical University of Denmark.
46. Tseng, C-T., Liao, C-J. (2008).A discrete particle swarm optimization for lot-streaming flowshop scheduling problem. European Journal of Operational Research, 191(2):360–373.
47. Tsiligirides, T. (1984). Heuristic methods applied to orienteering. Journal of the Operational Research Society, 35(9):797–809.
48. Van den Bergh, F., Engelbecht, A.P. (2000). Cooperative learning in neural networks using particle swarm optimizers. South African Computer Journal, 26:84–90.
49. Venter, G., Sobieski, J. (2002). Particle swarm optimization. 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Material Conference, Denver, CO.
50. Wang, Q., Sun, X., Golden, B. L., & Jia, J. (1995). Using artificial neural networks to solve the orienteering problem. Annals of Operations Research, 61:111–120.
51. Wang, X., Golden, B., & Wasil, E. (2008). Using a genetic algorithm to solve the generalized orienteering problem. In: Golden, B., Raghavan, S., Wasil, E. (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges, 263–274.
52. Yeh, L. W. (2003). Optimal procurement policies for multi-product multi-supplier with capacity constraint and price discount. Masters Thesis, Yuan Ze University Taiwan.

BIOGRAPHICAL SKETCH

Shanthi Muthuswamy is an Assistant Professor in the Department of Technology at Northern Illinois University. She received her Ph.D. in Industrial and Systems Engineering from State University of New York at Binghamton. Her teaching and research interests include heuristic optimization, facilities planning, system simulation, project management, and manufacturing systems.



Sarah Lam is an Associate Professor in the Systems Science and Industrial Engineering Department and an Assistant Director of Systems Analysis and Modeling of the Watson Institute for Systems Excellence (WISE), at the State University of New York at Binghamton. She received an M.S. degree in operations research from the University of Delaware and a Ph.D. degree in industrial engineering from the University of Pittsburgh. Her current research involves modeling and simulation, evolutionary optimization, data mining, and neural network modeling and validation. She is a member of IIE and IEEE.