An Improved Particle Swarm Optimization for the Multi-Depot Vehicle Routing Problem

Wenjing Zhang, Jianzhong Ye
Department of Computer Science and Technology
East China Normal University
Shanghai, China
wenjingzhang1214@126.com, jzye@cs.ecnu.edu.cn

Abstract—**This paper proposes a formulation of the multi-depot vehicle routing problem (MDVRP) that is solved by the particle swarm optimization (PSO) algorithm. PSO is one of the evolutionary computation technique, motivated by the group organism behavior such as bird flocking or fish schooling. Compared with other search methods, such as genetic algorithm, ant colony optimization and simulated annealing algorithm, PSO has many advantages like only primitive mathematical operators, high precision and fast convergence. However, it may premature and trap into the local optima sometimes. In order to overcome the drawback, this paper introduces a modified PSO algorithm with mutation operator and improved inertia weight. The simulation results shown that this modified method could not only avoid premature automatically according to the convergence level but also get a better optimal solution than the basic one.**

Keywords- multi-depot vehicle routing problem; particle swarm optimization; mutation operator; inertia weight

I. INTRODUCTION

The vehicle routing problem (VRP) is a well known combinatorial optimization problem for the sake of serving a number of customers with a fleet of vehicles. Firstly proposed in 1959, VRP is an important problem in the fields of transportation, distribution and logistics[1]. Assumed that the number, the position, the need of the customers and the loading capacity of the vehicles are all known, under various constraints, the objective of VRP is to find a suitable routing plan with the shortest total travel distance or the minimum time and so on. This is a very important studying field in operation research and is one of the representative combinatorial optimization problems.

This paper focuses on the multi-depot vehicle routing problem (MDVRP) that has multiple depots and every depot has at least one vehicle. It has been proved that VRP is NP-hard[2]. As a variant of VRP, MDVRP is also an NP-hard problem. It is difficult to solve this kind of NP-hard problem with accurate algorithms, therefore heuristic algorithms are widely used. Especially with the fast development of biology and physics, more and more useful algorithms that developed from these fields are applied to solve the VRP. These algorithms include genetic algorithm, taboo search, simulated annealing, ant colony optimization and particle swarm optimization (PSO). Compared with other algorithms, PSO has only primitive mathematical operators, high precision and fast convergence. [3] proposed a path planning algorithm using the particle swarm optimization and the improved Dijkstra algorithm for the mobile robot. In paper [4] a algorithm that integrated niche technology with the algorithm of PSO was used for vehicle routing and scheduling problem; the inertia weight decreased linearly to improve the searching ability. [5] proposed a PSO algorithm with multiple social structures for solving the VRP with simultaneous pickup and delivery. In paper [6] a fixed algorithm that combined PSO with simulated annealing was designed for solving the VRP with simultaneous delivery and pick-up. Different from these methods above, this paper designs an improved PSO algorithm that could vary particles dynamically according to the convergence degree.

The remainder of this paper is organized as follows. We give a detailed description of the MDVRP and the mathematical model in Section II. In Section III we state the standard PSO and two aspects of the improved strategy about PSO. The computational results are shown in Section IV and the conclusion is drawn in Section V.

II. PROBLEM DESCRIPTION AND MATHEMATICAL MODEL

The MDVRP can be described briefly as follows. Considering a routing network, it is represented by an undirected graph $G = (V, E)$, where V is a vertex set and E is an arc set. The subset $V_c = \{V_1, V_2, ..., V_n\}$ is the customer vertex set. Another subset $V_d = \{V_{n+1}, V_{n+2}, ..., V_{n+m}\}$ is the depot vertex set. The objective of this problem is to find the optimal distribution of vehicles by comparing the costs of different traveling plans. The DMVRP must satisfy the constraints as following:

- Every customer could be served arbitrarily by a vehicle and must be visited exactly once by only one vehicle.
- Every vehicle must start from and return to the same depot.
- The Vehicle can not start from a depot and go directly to the other one.
- The total demand of all the customers can not exceed the loading capacity of the vehicle that provides service for them.

978-0-7695-3997-3/10 $26.00 © 2010 IEEE
DOI 10.1109/ICEE.2010.803

Supposed that there are N customers that denoted by 1, 2, 3, ...N; M is the total number of the vehicle, so vehicles could be denoted by N+1,N+2, N+3, ..., N+M; D_{ij} is the distance between i and j; q_{mk} is the loading capacity of vehicle k in depot m; K_m is the number of vehicles in depot m; g_i is the need quantity of customer i; $x_{ij}^{mk} = 0$ if vehicle k in depot m starts from vertex i to j, otherwise $x_{ij}^{mk} = 0$.

As mentioned in paper [7], the mathematical model of MDVRP is constructed as following:

$$Min = \sum_{i=1}^{N+M}\sum_{j=1}^{N+M}\sum_{m=1}^{M}\sum_{k=1}^{K_m} d_{ij}\, x_{ij}^{km} \tag{1}$$

$$\sum_{i=1}^{N} g_i \sum_{j=1}^{N+M} x_{ij}^{mk} \le q_{mk} \tag{2}$$

$$m \in \{N+1, N+2, \cdots N+M\} \quad k \in \{1,2,\cdots K_m\}$$

$$\sum_{j=1}^{N}\sum_{k=1}^{K_m} x_{ij}^{mk} \le K_m \tag{3}$$

$$i = m \in \{N+1, N+2, \cdots N+M\}$$

$$\sum_{j=1}^{N} x_{ij}^{mk} = \sum_{j=1}^{N} x_{ji}^{mk} \le 1 \tag{4}$$

$$m \in \{N+1, N+2, \cdots N+M\} \qquad k \in \{1,2,\cdots K_m\}$$

$$\sum_{j=1}^{N+M}\sum_{m=1}^{M}\sum_{k=1}^{K_m} x_{ij}^{mk} = 1 \quad i \in \{1,2\cdots N\} \tag{5}$$

$$\sum_{i=1}^{N+M}\sum_{m=1}^{M}\sum_{k=1}^{K_m} x_{ij}^{mk} = 1 \quad i \in \{1,2\cdots N\} \tag{6}$$

$$\sum_{j=N+1}^{N+M} x_{ji}^{mk} = \sum_{j=N+1}^{N+M} x_{ij}^{mk} = 0 \tag{7}$$

$$m \in \{N+1, N+2, \cdots N+M\} \quad k \in \{1,2,\cdots K_m\}$$

Formula (1) is the objective function that to minimize the total travel distance of all the vehicles. Constraint (2) promises that the demand of the customers do not exceed the loading capacity of the vehicle that provides service for the rout. Constraint (3) guarantees that the number of vehicles that depart from depot m is no more than K_m. (4) states that vehicles start from and return to the same depot. Formula (5) and (6) represent that a customer must be served only once. (7) means that a vehicle can not depart from a depot and go directly to the other one.

III. IMPROVED PSO FOR MDVRP

A. Basic PSO

PSO is a population based stochastic technique that was proposed in 1995[8]. As birds in nature fly around looking for food, particles of the PSO algorithm searching for the best solution that has the optimal fitness value. A particle's status is defined by two factors: position and velocity. If the swarm with N particles in a D-dimensional searching space, position and velocity of particle i can be described respectively as $X_i = (x_i^1, x_i^2, ..., x_i^D)$, $V_i = (v_i^1, v_i^2, ..., v_i^D)$. Affected by three factors i.e. its own velocity, $P_i = (p_i^1, p_i^2, ..., p_i^D)$ (best position of particle i has achieved) and $G = (g^1, g^2, ..., g^D)$ (best position of all the particles in the swarm), the particle changes its velocity based on its own searching experience as well as the searching experience of other particles.

The particle changes its position and velocity as the following equations:

$$v_i^d(Iter+1) = \omega * v_i^d(Iter) + c_1 r_1 * (p_i^d - x_i^d(Iter)) + c_2 r_2 (g^d - x_i^d(\text{Iter})) \tag{8}$$

$$x_i^d(Iter+1) = x_i^d(Iter) + v_i^d(Iter+1) \tag{9}$$

where $1 \le i \le N$, $1 \le d \le D$; $v_i^d(Iter)$ is the velocity of particle i in dimension d at iteration $Iter$; $x_i^d(Iter)$ represents the position of particle i in dimension d at iteration Iter; w is the inertia weight factor; $C1$ is the cognition factor that controls the impact of the particle's best position; $C2$ is the social learning factor that controls the impact of the swarm's best position; $r1$ and $r2$ are two random number uniformly distributed in [0, 1]. In this paper formula (1) is used as the fitness function to calculate the minimal travel distance as the fitness value f which is used to access the performance of the particle.

As in paper [9], steps of the PSO algorithm are: Firstly, initialize the particles with random position and velocity within the solution space; Secondly, compute the value of the fitness function f_i of particle i, If the current value of f_i is better than f_{P_i}, then replace f_{P_i} and P_i by f_i and the current position; Use the same way to obtain G. Thirdly, change position and velocity of the particle based on equation (8) and (9). Repeat above steps until the maximal iteration $Iter_{max}$ or the stopping criterion is met.

B. Mutation of the Particle

By using PSO algorithm in MDVRP, the searching process may trap into a local optima and show early local convergence. In this paper, a partial mutation method is proposed to avoid this phenomenon.

First, we have to estimate the convergence degree of the swarm. As in probability and statistics, standard deviation σ indicates the deviation level of the samples:

$$\sigma = \sqrt{\frac{1}{N}\sum_{I=1}^{N}(x_i - \bar{x})^2} \tag{10}$$

where $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$; N is the number of the samples. Based on (10) and the fitness function, a definition named the swarm fitness deviation is introduced as following:

$$\sigma = \sqrt{\frac{1}{N}\sum_{I=1}^{N}((f_i - \bar{f})/\Delta f)^2} \tag{11}$$

where $\Delta f = \max_{1\le i\le N}\{(\max_{1\le i\le N} f_i) - \bar{f}\}$; $\bar{f} = \frac{1}{N}\sum_{i=1}^{N} f_i$ is the average value of the fitness function of all particles in the swarm; Δf is the maximal difference in fitness value of all the swarm; $\max f_i$ represents the maximal fitness value of particle i.

Then, based on the convergence level of the swarm, choose the particles that to be varied. The value of σ not only

illustrates the convergence level of the swarm in current iteration but also is inversely proportional to the convergence degree of the swarm. A big value of σ represents the low convergence degree of the swarm. At the early stage, particles are decentralized, so the value of σ is big. With the development of the searching process, particles become more and more centralized and the value of σ becomes smaller and smaller. Especially, if the swarm traps into the local optimal, we should re-decentralize the particles to restart a global searching. The number of the particles that to be varied is $N*P$, where N is the number of the total particles; P is the ratio of the particles that to be changed. The following equation is used to calculate the value of P.

$$P = 1 - \sigma \tag{12}$$

Third, vary the chosen particles and the strategy that is utilized here is to vary X_i. We know that if the swarm in a high convergence degree, the value of σ is small. So calculated by formula (12), a big P is got. More particles would be varied using the following equation:

$$x_i^d = x_i^d (1 - r) \tag{13}$$

where r is a random number between [0, 1]. In order to keep the original character of the particles, half $N*P$ particles use (13) to vary the front half part [1, D/2] of X_i. The remainder half $N*P$ particles use (13) to change the half part [D/2, D] of X_i in the rear. Algorithm 1 shows the procedure of this mutation.

Algorithm 1. Mutation of X_i
Step 1: calculate the value of σ ;
Step 2: use formula (12) to gain the ratio P;
Step 3: choose $N*P$ particles to vary their X_i;
Step 4: half $N*P$ particles use formula (13) to change [1, D/2] of X_i, another half part use (13) to vary [D/2, D] of X_i.

C. Improvement of the Inertia Weight

The inertia weigh w is an important parameter that controls the global/local search behavior of the PSO algorithm[9]. It impacts the particle's previous velocity on current velocity's changing tendency. As mentioned in [10], a high inertia weight means that the particle tends to keep its current velocity and the cognitive and social experience have little influence; a big value of w also results in global exploration. On the contrary, the small w means that the cognitive and social experience have big impact, which leads the particle to escape the current direction and results in local exploitation. At the early stage of the searching process, relatively large w is better to be used, so the swarm could search globally to get the best personal and global solutions. At the final stage, by using small w, the swarm could search locally and have faster convergence velocity. Zhang et al. introduced a linearly decreasing method of the inertia weight. In [4] w decreases linearly according to the increase of the algorithmic iteration as shown as following:

$$w(Iter) = w_{\max} - (w_{\max} - w_{\min}) \frac{Iter}{Iter_{\max}} \tag{14}$$

where w_{max} and w_{min} is the initial and final value of w.

In this paper a non-linear decreasing function is proposed to reduce w. It will be changed by using the following formulas:

$$w(Iter) = \sqrt{w_{\max} - (w_{\max} - w_{\min}) \frac{Iter}{Iter_{\max}}} \tag{15}$$

$$w(Iter) = \left(w_{\max} - (w_{\max} - w_{\min}) \frac{Iter}{Iter_{\max}}\right)^2 \tag{16}$$

As in equation (14), w in (15) and (16) also decreases with the increasing of the iteration. Because formula (15) is a convex function, w decreases faster at the final stage of iteration than the beginning. For the sake of keeping the global searching ability of the swarm, most of the particles should use the big w as long as possible. In this situation, w with a smaller decreasing ratio is a better choice. So we pick up $N*(0.5(1+\sigma))$ particles to use equation (15) to change w. In formula (16), even the inertia weight decreasing as it in (15), but w decreasing slower at the final stage than the early stage. So the rest $N*(0.5(1-\sigma))$ particles use formula (16) to vary w. By this method, not only w decreases non-linearly as the increasing of the iteration but also its changing strategy is related to the value of σ which represents the convergence degree of the swarm.

Steps of the modified PSO are shown in Algorithm 2.

Algorithm 2. Modified PSO
Step 1: Initialize X_0 and V_0 of all the particles.
Step 2: Choose P_0 and G; Use formula (1) to compute $f(P_0)$ and $f(G)$.
Step 3: Loop the next steps until the maximal iteration $Iter_{max}$ or the stopping criterion is met.
Step 3.1: Use algorithm 1 to vary the particles.
Step 3.2: Use (15) and (16) to compute w.
Step 3.3: Update X_i and V_i.
Step 3.4: For every particle if $f(P_i) < f(P_{i-1})$, replace $f(P_{i-1})$ by $f(P_i)$; else just keep the original value.
Step 3.5: If $f(G)$ in current iteration is smaller than the old one, then update it; else just keep the old value.

IV. Computational Results

A computational experiment has been done to compare the performance of the basic and the modified PSO. Considering the case in [7], there are sixteen customers, three depots and each depot has two vehicles. Each customer has constant demand and each vehicle has constant loading capability. Parameters of the customers and the depots are shown in table I and II .

TABLE I. Customer data

Customer	X_ Coordinate	Y_ Coordinate	Demand
1	78	30	9
2	39	31	3
3	20	76	2
4	65	98	2
5	95	23	2.5
6	90	9	4.5
7	33	65	3.5
8	79	75	2.5
9	55	35	3.5
10	88	55	2.5
11	9	12	3
12	35	93	2.5
13	73	85	4

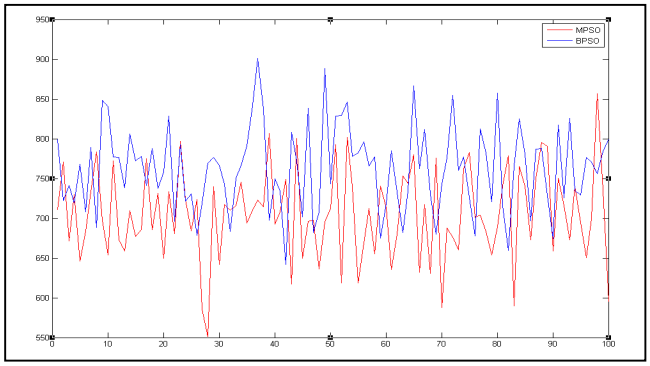
14	62	53	2
15	33	13	4
16	11	20	1.5

TABLE II. DEPOT DATA

Depot	***1***	***2***	***3***
X_coordinate	20	75	50
Y_coordinate	20	45	80
Vehicle_Num	2	2	2
Capacity	8/10	8/10	8/10

The algorithm is run under the Matlab 7.5. In this paper the swarm size is ten times of the customer, so the number of particle is 160; $w_{max} = 0.9$, $w_{min} = 0.1$; $c1 = 2$, $c2 = 2$; $Itre_{max} = 2000$.

By testing the basic PSO (BPSO) and the modified PSO (MPSO) algorithm 100 times respectively, the comparison of optimal solutions is shown in figure 1.



Fingure 1. Comparison of optimal solutions

From the two algorithms' 100 times test results that shown in figure 1, we can see that the red line representing the improved PSO almost below the blue line representing the basic PSO. So the improved PSO algorithm could always get better travel distance than the basic one.

Comparisons of the average optimal distance (A-O-D) and the deviation of both algorithms are shown in table III.

TABLE III. COMPARISONS OF A-O-D AND DEVIATION

Algorithm	**BPSO**	**MPSO**
A-O-D	762.6410	705.4236
Deviation	5.3190	5.6524

From table III, we could draw a conclusion that the average travel distance of the modified PSO decreases 7.5%. Generally, the improved PSO could gain better average optimal solution than the basic PSO. But the deviation value of the modified PSO is bigger than the basic one, which illustrates that the improved algorithm fluctuates more greatly than the basic one.

The best routing plans of the two algorithms that get from the 100 times test are shown in table IV.

TABLE IV. ROUTING PLANS

Depot	**Vehicle**	**customer**	
		BPSO	***MPSO***
1	***1***	11, 15	11, 16
	2	3, 7, 12, 16	7, 12, 3
2	***3***	4, 13, 14	9, 14, 10
	4	5, 10, 8	8, 13, 4
3	***5***	2, 9	2, 15
	6	1, 6	1, 5, 6

The optimal distance of basic and modified PSO respectively is 641.57 and 550.77, so the improvement ratio is 14.15%.

V. CONCLUSION

In this paper, the MDVRP is solved by the PSO algorithm. We improve the PSO algorithm from two aspects to avoid the local optima phenomenon. One is varying the position of the particle according to the convergence level of the swarm, which can avoid premature effectively. Another aspect is based on the convergence level of the swarm to improve the decreasing model of the inertia weight. From the experimental results we can see that the modified PSO could get better average optimal solution and better optimal routing plan than the basic PSO algorithm. But the modified PSO algorithm need further modification because of its big deviation value. The coding method in the computational test also need to be optimized.

ACKNOWLEDGMENT

The authors are grateful to Weinan Gao and Ting Chen for their constructive advice.

REFERENCES

[1] G. B. Dantzig and R. Ramser, "The Truck Dispatching Problem," Management Science, 1959.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithm, second edition, 2001.

[3] H. II. Kang, B. Lee, K. Kim, "Path Planning Algorithm Using the Particle Swarm Optimization and the Improved Dijkstra Algorithm," 2008 IEEE Pacific-Asia Workshop on Computation Intelligence and Industrial Application.

[4] Z. Zhang, C. Lu, "Application of the Improved Particle Swarm Optimizer to Vehicle Routing and Scheduling Problems," Proceeding of 2007 IEEE International Conference on Grey Systems and Intelligent Services.

[5] T. Ji Ai*, V. Kachitvichyanukul, "A Particle Swarm Optimization for the Vehicle Routing problem with Simultaneous Pickup and Delivery," Computer & Operation Research 36,2009.

[6] T. Zhang, C. Yu, "A Mixed PSO Algorithm for the VRPSPD," IEEE 2008 Chinese Control and Decision Conference.

[7] L. Wen, F. Meng, "An Improved PSO for the Multi-Depot Vehicle Routing Problem with Time Windows," 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application.

[8] J. Kennedy, R. Eberhart, "Particle swarm optimization," Proceeding of IEEE International Conference on Neural Networks, pp.1942-1948,1995.

[9] J. Kennedy, R. Eberhart, "A new optimizer using particle swarm theory," Proceedings of the Sixth Iinternational Symposium on Micro Machine and Human Science, pp.39-43, 1995.

[10] Y. Shi, R. Eberhart, "A Modified Particle Swarm Optimazer," IEEE International Conference on Evolutionary Compution, IEEE press, Piscataway,NJ, pp. 69-63,1998.