An Improved PSO for the Multi-Depot Vehicle Routing Problem with Time Windows

Lei Wen[1] , Fanhua Meng[2]
[1,2]*Business and Management department*
North China Electric Power University Baoding, Hebei, 071003, China
wenlei0312@sohu.com

Abstract

The distribution of finished products from depots to customers is a practical and challenging problem in logistics management. Better routing and scheduling decisions can result in higher level of customer satisfaction. The distribution problem is generally formulated as the vehicle routing problem (VRP). Nevertheless, there is a rigid assumption that there is only one depot. In cases, for instance, where a logistics company has more than one depot, the VRP is not suitable. To resolve this limitation, this paper focuses on the VRP with multiple depots with time windows, (MDVRPTW). The MDVRP is NP-hard, which means that an efficient algorithm for solving the problem to optimality is unavailable. To deal with the problem efficiently, improved PSO is developed in this paper. A computational study is carried out to verify the algorithms. It is proved that the performance of improved PSO can solve this problem efficiently.

1. Introduction

Vehicle routing problems (VRP) are well known combinatorial optimization problems, it was firstly proposed by Dantzing and Ramser in 1959 [1]. The vehicle routing problem (VRP) is the problem of minimizing the total travel distance of a number of vehicles, under various constraints, where every customer must be visited exactly once by a vehicle. This is one of the representative combinatorial optimization problems and is known to be NP-hard. There are many variants to vehicle routing problems. One such variant is the multi-depot vehicle routing problem with time windows (MDVRPTW). The objective of the problem is to find routes for vehicles to service all the customers at a minimal cost (in terms of number of routes and total travel distance), without violating the capacity and travel time constraints of the vehicles and the time windows imposed by the customers. MDVRPTW is a typical case of NP-hard problem, which is nearly impossible to find the accurate solution. it is usually to find the solution by using a group of heuristic algorithm or biological evolution algorithm when solving such a kind of problem. MDVRP have recently received an increasing amount of attentions and many research papers relevant to them have been published in the literature. Wu TH et al. [2] divided multi-depot location-routing problem into two sub-problems: location allocation and the general VRP. Each sub-problem is solved by a simulated annealing algorithm. Michael Wasner et al. [3] proposed a heuristic solution concept using four times of feedback to solve MDVRP. Maria et al. [4] defined an auxiliary network and utilized Tabu search to solve a combined location-routing problem. Kaoru Hirota et al. [5] decomposed the problem into three layers: Atomic layer, Molecular layer and Individual layer. After the initial routes were constructed, trip moves and exchange operations were performed between tours in order to improve the quality of solutions. Filipec M et al. [6] used three phases to solve multi-depot capacitated VRP. First, the customers were divided into regionally bounded clusters. Second, genetic algorithm (GA) was applied for each depot to optimize radial routings.

Different from the above methods, Improved PSO is applied in this paper. We first describe a practical problem and establish its mathematical model. Then, we state in detail the design of the algorithm for solving the problem. Finally the computational results of the actual instance show the effectiveness of the proposed method.

2. Problem description and mathematical model

The VRPTW can be described briefly as follows. Let us consider a routing network, represented by the directed graph G {V,E} where V is a vertex set, and E- $(V_i, V_j)$is an arc set. V includes two subsets: $V_c = \{V_1, V_2, \cdots, V_n\}$ is a customer vertex set and $V_d = \{V_{n+1}, V_{n+2}, \cdots, V_{n+m}\}$ is a depot vertex set. For each $V_i \in V_c$, it includes the demand of the customer; for each $V_i \in V_d$, it includes the number of the vehicle where K_{v_i} stands for the number of the vehicle; C_{ij} is the transportation cost on link (V_i, V_j).

978-0-7695-3490-9/08 $25.00 © 2008 IEEE
DOI 10.1109/PACIIA.2008.9

The MDVRPTW consists of designing a set of least cost vehicle routes such that:

- Every route starts and ends at the depot.
- Every customer of V_i (excluding the depot) is visited exactly once by exactly one vehicle.
- The total demand of any vehicle route does not exceed the vehicle capacity.
- Each customer V_i has a required service time
- Each customer V_i has a service time window $[ET_i, LT_i]$ where ET_i is the earliest time that service, can begin and LT_i is the latest time that service can begin. Each delivery can be done no later than the ending time of the customer's time window, while if the arrival time at the customer's location is before the beginning of the Customer's time window, the delivery has to wait until the beginning of the time window.

Two types of time window constraints are known as a hard time window and soft time window.

- hard time window

The vehicle must start the customer service after ET_i and before LT_i. The concrete constraining formula expressed as: $P_i = M \times \max(ET_i - T_i, 0) + M \max(T_i - LT_i, 0)$ Where P_i the penalty cost and M is the penalty coefficient being equal to $+\infty$.

- soft time window

Soft time window, that is, if the arrival time at a customer is later than the end of the time window, the cost function (the total travel time here) will be penalized by some amount. The penalty function expressed as:

$P_i = a \times \max(ET_i - T_i, 0) + b \max(T_i - LT_i, 0)$

Where P_i is the penalty cost and a and b are the penalty coefficient. In this paper we use the soft time window.

In order to simplify the problem, we define the sets, parameters and variables used in the mathematical model as follows:

We denote the customers by 1,2,…,N and depots by N+1,N+2,…,N+M .

variables:

$x_{ij}^{mk} = \{1, 0\}$ If vehicle k in the depot m travels from vertex i to vertex j then x_{ij}^{mk} =1, otherwise x_{ij}^{mk} =0.

T_i arrival time at node i

N total number of customers

M total number of depots

C_{ij} cost incurred on arc from node i to j

K_m total number of vehicles in depot m

ET_i earliest arrival time at node i

LT_i latest arrival time at node i

g_i demand at customer i

q_{mk} capacity of vehicle k in depot m

The formulation of the problem is as follows:

$$\min = \sum_{i=1}^{N+M}\sum_{j=1}^{N+M}\sum_{m=1}^{M}\sum_{k=1}^{K_m} C_{ij} x_{ij}^{mk} + a \times \sum_{i=1}^{n} \max(ET_i - T_i, 0) + b \sum_{i=1}^{n} \max(T_i - LT_i, 0) \quad (1)$$

$$\sum_{j=1}^{N}\sum_{j=1}^{K_m} x_{ij}^{mk} \le K_m \quad (2)$$

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

$$\sum_{j=1}^{N} x_{ij}^{mk} = \sum_{j=1}^{N} x_{ij}^{mk} \le 1 \quad (3)$$

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

$$\sum_{j=1}^{N+M}\sum_{m=1}^{M}\sum_{k=1}^{K_m} x_{ij}^{mk} = 1, i \in \{1, 2, \cdots, N\} \quad (4)$$

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

$$\sum_{i=1}^{N} g_i \sum_{j=1}^{N+M} x_{ij}^{mk} = \le q_{mk} \quad (6)$$

$$m \in \{N+1, N+2, \cdots N+M\} \; k \in \{1, 2, \cdots, k_m\}$$

$$\sum_{j=N+1}^{N+M} x_{ji}^{m+k} = \sum_{j=N+1}^{N+M} x_{ij}^{m+k} = 0 \quad (7)$$

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

The objectives, formula (1) is to minimize the total vehicle travel cost; Constraint (2) limits the number of vehicles going out of the depot m less than K_m; Formula (3) guarantees that each vehicle sets out from the depot and returns to it; Constraint (4) and (5) ensure that a customer is serviced exactly once by only one vehicle; Formula (6) constrains the vehicle capacity and formula (7) ensure that that the vehicles could not be from the depot to depot.

3. Improved PSO for MDVRPTW

3.1 Stander PSO

Particle Swarm Optimization (PSO), an evolutionary computation technique firstly introduced by Kennedy and Eberhart in 1995 [7, 8], is an optimization algorithm based on swarm theory. The main idea of PSO is to model the flocking of birds flying around a peak in a landscape. A particle's status on the search space is characterized by two factors: its position and velocity. The position and velocity of the *i*th particle in the *d*-dimensional search space can be represented as $X_i = (x_{i1}, x_{i2}, x_{i3} \cdots x_{iD})$ and $V_i = (v_{i1}, v_{i2}, v_{i3} \cdots v_{iD})$, respectively. Each particle has its own best position $P_i = (p_{i1}, p_{i2}, p_{i3} \cdots p_{iD})$

corresponding to the personal best objective value obtained so far at time *k*. The global best particle is denoted by $P_g = (p_{g1}, p_{g2}, p_{g3} \cdots p_{gD})$, which represents the best particle found so far at time 可. The new velocity of each particle is calculated as follows:

$$v_{id}^{k+1} = wv_{id}^{k} + c_1 r_1 * (p_{id} - x_{id}^{k}) + c_2 r_2 * (p_{gd} - x_{id}^{k}) \quad (8)$$

Where *c*1 and *c*2 are constants called acceleration coefficients, *w* is called the inertia weight factor, *r*1 and *r*2 are two independent random numbers uniformly distributed in the range of [0, 1]. Thus, the position of each particle is updated in each generation according to the following equation:

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \qquad 1 \le i \le n; 1 \le d \le D \quad (9)$$

Generally, the value of each component in v_{id} by (8) can be clamped to the range $[-V_{\max_d}, V_{\max_d}]$ to control excessive roaming of particles outside the search space. Then the particle flies toward a new position according to (9). This process is repeated until a user-defined stopping criterion is reached.

3.2 Improved Method

Inertia weight decides how much velocity the particle inherits from the current speed. The empirical studies of PSO with inertia weight have shown that a relatively large *w* has more global search ability while a relatively small *w* results in a faster convergence. Linearly decreasing inertia weights were recommended by the authors Zheng et al. [9] claimed that PSO with declining inertia weight performs better. In this paper, we propose a new method to change the Inertia weight with nonlinear time-decreasing function. Proper selection of inertia weight gives balance between global and local searching [10]. In this paper, we propose a new method to change the inertia weight with nonlinear time-decreasing function. We designed 2 the nonlinear time-decreasing inertia weight function.

$$y = \sqrt{1-x}, \quad (0 \le x \le 1) \; x = iter / Maxiter \quad (10)$$

here *iter* denoted current iteration times, *Maxiter* denoted the acceptable max iteration times. This function is a convex function. The inertia weight will declined with the iteration increased. at the beginning, the inertia weight decreasing slowly, at the end stage the inertia weight decreasing faster.

$$y = (1-x)^2, \quad (0 \le x \le 1) \; x = iter / Maxiter \quad (11)$$

this function is a concave nonlinear function, which decreases faster at the beginning than in the last stage.

Through some experiments, we can find that the nonlinear time-declining inertia weight performs better than constant one. These functions will be respectively applied to certain particles.

3.3 Particle Definition and Solution Expression

In this section, we describe the formulations of the PSO algorithm for MDVRPTW. In this paper, , we assume that processors directly connected with each other. We apply the PSO as if they were searching a continuous space, and then round the positions (routes) to integer values. In our work, we set up a new particle coding for the MDVRPTW which helps convert the discrete combinational problem into continuous problem so that the improved PSO algorithm can be directly applied. This paper constructs a 2 L-dimensional vector corresponding to L service points MDVRPTW, each service point includes two-dimensional vector: the number of service vehicle and the order in which they are visited. To facilitate expression and calculated, each corresponding particle includes 2 L-dimensional vector X which is divided into two L-dimensional vector: X_v and x_{vi}. For example, suppose that there are seven customers numbered 1–7 and three vehicles in the depot. If the particle code representation is,

x_{vi} : 1 2 2 2 2 3 3 x_{ri} : 1 3 5 4 6 7 2

Then three routes are required to serve all these seven customers. In the first route, vehicle 1 starts from the depot, and travels to customers 1.After that, the vehicle returns back to the depot. In the second route, the vehicle 2 starts with customer 3, then customer 5, 4, and finally customer 6. Similarly, the vehicle 2 travels back to the depot after serving the customers. In the third route, the vehicle 3 starts with customer 7, then customer 2. Similarly, the vehicle 3 travels back to the depot after serving the customers. Note that each particle contains 2n links if there are n depots in the MDVRPTW. In this paper, the single depot coding method extends to multi-depot coding method, being equal to the problem contains some depots putting the particle according to the above coding method and expanding into several parts in the middle to distinguish between 0. For example, there being n depots, the coding method can be expressed:

$$X_v = [x_{v1,}\, 0, x_{v2,}\, 0, \cdots 0, x_{vn,}], \quad X_r = [x_{r1,}\, 0, x_{r2,}\, 0, \cdots 0, x_{rn,}]$$

3.4 Initialization and local adjustment

(1) Initialization

We should determine some initial feasible solutions; this will be conducive to faster and more accurate algorithm to find the optimal solution found. In the stage of initialization, there are three steps to generate a feasible initial solution. The first step is to assign customers to each of depot that is the grouping problem. Because the objective here is to minimize the total delivery cost i.e. the total distance in distribution, customers are assigned to the nearest depot. For example, there are two depots, that is, d_A and d_B available in the MDVRPTW. Each customer, say V_i, should be assigned to a single depot exactly. The

selection is based on the following calculation:

If $D(V_i, d_A) < D(V_i, d_B)$, then assign V_i to d_A;

If $D(V_i, d_A) > D(V_i, d_B)$, then assign V_i to d_B;

In case $D(V_i, d_A) = D(V_i, d_B)$, select a depot arbitrarily.

Where $D(V_i, d_k) = \sqrt{(x_{V_i} - x_{d_k})^2 + (y_{V_i} - y_{d_k})^2}$. The second step is to assign customers in the same link to several routes using the Clarke and Wright (1964) saving method. The saving regarded in this paper is the distance traveled by the vehicles for serving the customers. The method is to construct a saving matrix, $S(V_i, V_j)$, for every two customers in the same link first. Then, the customers with larger saving value are grouped in the same route while not violating the vehicle capacity constraint.

Saving matrix for Link A:

$S(V_i, V_j) = D(d_A, V_i) + D(d_A, V_j) - D(V_i, V_j)$;

Saving matrix for Link B:

$S(V_r, V_s) = D(d_B, V_r) + D(d_B, V_s) - D(V_r, V_s)$;

The third step is to solve the scheduling problem by the NNH (Reinelt, 1994). The principle of the NNH is to start with the first customer randomly, then to select the next customer as close as possible to the previous one from those unselected customers to form the delivery sequence until all customers are selected.

(2) Local adjustment

Local adjustment can switch the customers between the closet depot and the second closet depot if the ratio of the distances to the closest and second closest depots more than one setting value λ. Specific steps and methods are as follows:

Step1: Establish a neighboring depots table NearD=Null. For every $V_i \in V_c$, we calculate the distance from it to the closet depot D_i' and second closet depot D_i''. The two distances were marked respectively $C_{V_iD_i'}$ and $C_{V_iD_i''}$. First, assign V_i to D_i', meaning to V_i is serviced by D_i'. Then, put $r_i = C_{V_iD_i'} / C_{V_iD_i''}$, if r_i is more than one setting value λ (λ≤1), put V_i into the neighboring depots table $NearD = NearD \cup V_i$. All the customers in the neighboring depots table can be exchange between the closet depot and second closet depot. Repeated the operation until all V_i a traversal, then V_i are assigned to a depot and determine the depots that can be exchange between the closet depot and the second closet depot.

Step2: Calculate the number of the table NearD and mark the number with NearNum;

Step3: Randomly generated a number i = random (1 ~ NearNum);

Step4: P=NearD [i], find its closet depot Near=Nearest [P], second closet depot S=Second [P] and the current depot N=Now[P]

Step5: In particle r, remove P;

Step6: If N=Near, P will inserted into any position in X_{rs} and make Now[P]=S.If N=S, P will inserted into any position in X_{rs} and make Now[P]=S.

3.5 The procedure for the implementation of PSO

First, we modify the updating formula in PSO, because we use the real-coded method, based on the updated velocities, each individual (particle) changes its position according to the equations as follows:

$$v_{id}^{k+1} = \text{int}\{wv_{id}^k + c_1 r_1 * (p_{id} - x_{id}^k) + c_2 r_2 * (p_{gd} - x_{id}^k)\} \quad (12)$$

$$x_{id}^{k+1} = Abs\{x_{id}^k + v_{id}^{k+1}\} \quad (13)$$

Here, Abs ( ) is the absolute value function. int ( ) is the integral function. r_1 and r_2 are two independent random numbers uniformly distributed in the range of [0, 1].

The process for implementing the improved PSO is as follows:

Step 1: Set the maximum of iterations taken $S_{\max}$ and Initialize the PSO parameters, c_1, c_2 .Initialize a population of particles as follows:

1.1: Generate the particles randomly. Initialize the particles using 3.1.

1.2: Calculate fitness values for all the particles and select the best position P_i of particle i and the global best position P_g of particles.

Step 2: loop to the next steps and repeat until a criterion is met or reach the $S_{\max}$.

2.1: we randomly select $\rho = 50\%(S_{\max} - t)/S_{\max}$ particles to update using the f formula (10) and the rest using formula (11).

2.2: update the particles

2.3: local adjustment

2.4: Calculate the new fitness function for all the particles' new positions. Determine the new P_i. Compare with the previous P_i and update the value with the new one if necessary. Calculate the new group best position P_g among all the new P_i. Compare with the previous P_g and update the group best before the next iteration.

4. Calculation example

A computational experiment has been conducted to analyze the performance of the proposed methodology. The computational experiment assumed to have three delivery depots and sixteen customers. Each customer has the earliest and latest allowable service time. Each vehicle has a constant loading capacity. Time and distance can be

converted in equal units, and the amount of each customer's demand is known. The concrete data can be seen in the table 1 and table 2.

Table.1 Customer data

customer	1	2	3	4	5	6	7	8
X coordinate	78	39	20	65	95	90	33	79
Y coordinate	30	31	76	98	23	9	65	75
demand	3	3	2	2	2.5	4.5	3.5	2.5
Service time	3.5	3	0.5	2	3	1	2.5	1.5
Time window	[2，5]	[4，7]	[2，5]	[1.5，4.5]	[5，8]	[1，4]	[3，6]	[5，7.5]

customer	9	10	11	12	13	14	15	16
X coordinate	55	88	9	35	73	62	33	11
Y coordinate	35	55	12	93	85	53	13	20
demand	3.5	2.5	3	2.5	4	2	4	1.5
Service time	2	1.5	1.5	2	3	1	2.5	0.5
Time window	[2，4]	[4，6]	[0.5，4]	[2，4]	[3，7]	[3，6]	[1，2]	[3，5]

Table.2 Depot data

depot	17	18	19
X coordinate	20	75	50
Y coordinate	20	45	80
Vehicle NO	2	2	2
capcity	8t / 10t	8t / 10t	8t / 10t

The algorithm is implemented under the Matlab 7.0. In order to avoid local optimum convergence, the number of the particles must be increased according to the size of the service. Basically, for a median size of the customers the number of the particles should be no less than 10 times of the customers. So the number of particles is 170. Neighbors subgroup size is 4; w_{int} =0.94, w_{end} =0.13 $c_1 = c_2$ =1.28, $S_{\max}$ =800. We can get a satisfactory result with the algorithm which has been validated by many times experiment. Decode the particle to get the optimal solution. The code of the particle is as follow:

$X_v = [1,1,1,2,2,0,1,1,2,2,2,0,1,1,1,2,2,2]$

$X_r = [11,16,2,15,6,0,9,14,1,5,10,0,12,3,7,4,13,8]$

This particle corresponds to the optimal solution is as follow:

Depot 1: Vehicle 1(8t): 17-11-16-2-17
Vehicle 2(10t): 17-15-6-17
Depot 2: Vehicle 1(8t): 18-9-14-18
Vehicle 2(10t): 18-1-5-10-18
Depot 3: Vehicle 1(8t): 19-12-3-7-19
Vehicle 2(10t): 19-4-13-8-19
The total length: 440.92

5. Conclusions

In the industrial business today, the optimal selection of the vehicle route is a key issue to improve the service quality, reduce the operation cost and increase the profits. In this paper a computational-efficient PSO algorithm is introduced to solve the MDVRPTW problem. The particle expression and the approach of the whole algorithm are detailed. The effectiveness of the algorithm is demonstrated by the experimental results.

References

[1] G.B.Dantzig and R.Ramser，“The Truck Dispatching Problem,” Management Science,vol.6, p.8091,1959.

[2] Wu TH, Low C and Bai JW, “Heuristic solutions to multi-depot location-routing problems”, Computers and operations Research, 2002, 29 (10), 1393-1415.

[3] Michael Wasner and Günther Zäpfel, “An integrated multi-depot hub-location vehicle routing model for network planning of parcel service”, Int. J Production Economics 90 (2004) 403-419.

[4] Maria Albareda-Sambola, Juan A. Díaz and Elena Fernández, “A compact model and tight bounds for acombined location-routing problem”, Computers & Operations Research, 2005, 32, 407–428.

[5] Hirota K, Chen KW and Dong FY, “Computational intelligence approach to real-world cooperative vehicle dispatching problem”, Second IEEE International Conference on Intelligent Systems, June 2004, 7-12.

[6] Filipec M, Skrlec D and Krajcar S, “ Darwin meets computers: New approach to multiple depot capacitated vehicle routing problem”, Proceeding of International Conference on System, Man, and Cybernetics, Orlando, USA, 1997, October, 12-15,421-426.

[7] J. Kennedy, R.C. Eberhart, Particle Swarm optimization, in: Proc. of IEEE Int. Conf. on Neural Networks, Perth, Australia (1995) 1942-1948.

[8] J. Kennedy, R.C. Eberhart, A new optimizer using particle swarm theory, in: Proc. of the Sixth Int. Symp. On Micro Machine and Human Science (MHS’ 95), Nagoya, Japan (1995) 39-43.

[9] Zheng, Y., Ma, L., hang, L., and Qian, J. On the convergence analysis and parameter selection in particle swarm optimization. Proceedings of International Conference on Machine learning and Cybernetics, 2003. pp. 1802-1807.

[10] X. Yang et al., A modified particle swarm optimizer with dynamic adaptation, Appl.Math.Comput.(2007), doi:10.1016/ j.amc.2006.12.045