SOLVE THE VEHICLE ROUTING PROBLEM VIA A DISCRETE PARTICLE SWARM OPTIMIZATION

***Hong-Wei Liu*[1]**

[1]*Information School, Beijing Wuzi University, Beijing 101149, China*
ryuhowell@163.com

Keywords: Capacitated vehicle routing problem, particle swarm optimization, NP-hard problem, optimal splitting procedure, benchmark instance.

Abstract

This paper introduces a new hybrid algorithmic approach based on discrete Particle Swarm Optimization (PSO) and optimal splitting procedure for solving one of the most popular supply chain management problems, the Vehicle Routing Problem (VRP). The VRP is a well known NP-hard problem in which a vehicle with finite capacity leaves from the depot with full load and has to serve a set of customers whose demands are known only when the vehicle arrives to them. Experiment results show that the proposed algorithm is an efficient algorithm in solving VRP. This paper also shows that the proposed algorithm algorithm can find the feasible solution effectively and almost can find the global optimal solution for small instance. For larger instances, if the size of populations in the proposed algorithm increases, the possibility of finding the global optimal solution will increase.

1 Introduction

In the fields of transportation, distribution, and logistics, the Vehicle Routing Problem (VRP) or the Capacitated Vehicle Routing problem (CVRP) arises naturally as a central problem in which vehicles based on the central depot are required to visit geographically dispersed customers exactly once in order to fulfill known customer demands. The VRP is often defined on a undirected network $G = (V;E)$ with a node set V and an edge set E. Node 0 is a depot and each other node $i > 0$ represents a customer and each edge has a non-negative travel cost. The reader can find more detailed descriptions of the algorithms proposed for the CVRP and its variants in the survey papers [1-8] and in the books [9-11].

Due to the fact that the vehicle routing problem is a well-known integer programming problem which falls into the category of *NP-Hard* problems, the instances with a large number of customers cannot be solved in optimality within reasonable time. For this reason, a number of approximation techniques (metaheuristic and evolutionary algorithms) were proposed for the solution of the problem. In this paper, the *Vehicle Routing Problem* is solved using one of these algorithms, the Particle Swarm Optimization algorithm. Particle Swarm Optimization (PSO) was originally proposed by *Kennedy* and *Eberhart* [12-14] to simulate the social behavior of social organisms such as bird flocking and fish schooling. Its mechanism enhances and adapts to the global and local exploration and this method has been identified as very useful in many problems. Most applications of PSO have concentrated on the optimization in continuous space while some work has been done to the discrete optimization [13,15]. Recent complete surveys for the Particle Swarm Optimization can be found in [16-19].

The most commonly used techniques for solving VRP are heuristics and metaheuristics because of the NP-Hardness of the problem when the number of cities is large. There exists 2-stage algorithm, Cluster-First-Route-Second Algorithms, and Route-First-Cluster-Second Algorithms in literature. Route-First-Cluster-Second methods construct in a first phase a giant TSP tour, disregarding side constraints, and decompose this tour into feasible vehicle routes in a second phase. This idea applies to problems with a free number of vehicles. The goal of this paper is to integrate and implement discrete PSO algorithm with Prins' mtehod [20] to handle VRP since there has little research on computational experience showing that Route-First-Cluster-Second heuristics are competitive with other approaches.

2 Methodology

2.1 General description of PSO

Particle Swarm Optimization (PSO) is a population based swarm intelligence algorithm that simulates the social behavior of social organisms by using the physical movements of the individuals in the swarm. In PSO algorithm, initially a set of particles is created randomly where each particle corresponds to a possible solution. Each particle has a position in the space of solutions and moves with a given velocity. The position of each particle is represented by a p-dimensional vector in problem space $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$, $i = 1, 2, \ldots, N$ (N is the population size and p is the number of the vector's dimension), and its performance is evaluated on the predefined fitness function ($f(x_{ij})$). The velocity (v_{ij}) represents the changes that will be made to move the particle from one position to another. Where the particle will move depends on the dynamic interaction of its own experience and the experience of the whole swarm. There are three possible directions that a particle can follow: to follow its own path, to move towards the best position it had during the iterations ($pbest_{ij}$) or to move to the best particle's position ($gbestj$).

In classical PSO algorithm, the position of a particle changes using the following equation:

$$\begin{cases} v_{ij}(t+1) = wv_{ij}(t) + c_1 rand_1(pbest_{ij} - x_{ij}(t)) + c_2 rand_2(gbest_j - x_{ij}(t)) \\ x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \end{cases} \quad (1)$$

where t is the iterations' counter. where c_1 and c_2 are the acceleration coefficients, $rand_1$ and $rand_2$ are two random variables in the interval [0, 1] and w is inertia weight. The acceleration coefficients c_1 and c_2 control how far a particle will move in a single iteration. Low values allow particles to roam far from target regions before being tugged back, while high values result in abrupt movement towards, or past, target regions [12]. A particle's best position ($pbest_{ij}$) in a swarm is calculated from the equation:

$$pbest_{ij} = \begin{cases} x_{ij}(t+1), & f(x_{ij}(t+1)) < f(x_{ij}(t)) \\ pbest_{ij}, & otherwise \end{cases}$$

The optimal position of the whole swarm in the VRP at time t is calculated by the equation:

$$gbest_j = \min\{f(pbest_{1j}), f(pbest_{2j}), \ldots, f(pbest_{Nj})\}.$$

The search mechanism of the PSO is demonstrated in **Figure** 1.

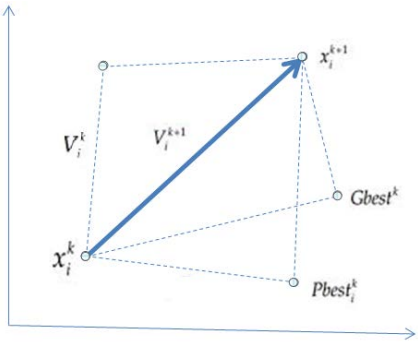

Figure 1. The search mechanism of the particle swarm optimization.

2.2 Solution representation

One of the key issues in designing a successful PSO for the Vehicle Routing Problem is to find a suitable mapping between Vehicle Routing Problem with solutions and particles in PSO. Each particle is recorded via the path representation of the tour without trip delimiters, that is, via the specific sequence of the nodes. It can be interpreted as the order in which a vehicle must visit all customers, assuming the same vehicle performs all trips in turn. This simple encoding is appealing because there obviously exists one optimal sequence. A tour may be broken into severaldifferent routes. For example, a tour with 8 customers, *S=(1, 2, 3, 4, 5, 6, 7, 8)* may be broken into *R1=(1,2), R2=(3, 4, 5), R3=(6,7,8)* or *R1=(1,2, 3), R2=(4,5,6)*and *R3=(7,8),etc.*. Christian Prins [20] proposed an optimal splitting procedure to get the best solution (among all the possible routes) respecting to a given tour. We will review the main idea of this splitting procedure in Section 2.3 and discuss our implementation in Section 2.4.

2.3 Calculation of the fitness function.

Because a tour can be split into many different routes, Prins [20] proposed a splitting procedure which can find an optimal splitting i.e a trip delimiter so that the total cost is minimized. The main idea can be described as follows. Without loss of generality, let $S = (1,2,3,\cdots,n)$ be a given tour. Consider an auxiliary graph $H = (\bar{V}, \bar{E})$ where $\bar{V} = \{0,1,2,\cdots,n\}$. An arc $(i, j) \in \bar{E}$ if

$$\bar{E}_{ij} = c_{0,i+1} + \sum_{k=i+1}^{j-1}(d_k + c_{k,k+1}) + d_j + c_{j,0}$$

$$s.t. \quad \sum_{k=i+1}^{j} q_k \le Q$$

Then $\bar{E}_{ij}$ is the total travel cost(time) for the route $(i+1,\ i+2,\cdots,\ j)$. An optimal split for S corresponds to shortest path *P* from vertex *0* to vertex *n* in *H*. The following example (Prins, [20]) demonstrates the construction. The first graph of Fig. 1 shows a sequence *S=(a, b, c, d, e)*. The number associates with each edge is the travel time. Let us assume the service time (delivery time) is 0 and the demands are 5, 4, 4, 2 and 7, respectively. The capacity of the vehicle is 10. Then the auxiliary graph *H* is the second graph in Fig. 1. The edge in *H* with weight 55 corresponds to the travel time of the trip *(0, a, b, 0)*. The weight associated with the other edges are similarly defined. The shortest path from 0 to e is *(0, b, c, e)* with minimal total travel time 205. The last graph gives the optimal result with three trips.

The auxiliary graph helps us understand the idea how to split a given tour *S* into optimal trips. But in practice, we do not have to construct such graph *H*. It can be done by a labeling algorithm and a splitting procedure [20]. Let *S = (1, 2,.., n)* be a given tour. Two labels V_j and P_j for each vertex *j* in *S* are computed. V_j is the cost of the shortest path from node *0* to node *j* in *H*, and P_j is the predecessor of *j* on this path. The minimal cost is given at the end by V_n. For any given *i*, note that the increment of *j* stops when *Q* are exceeded. The labeling algorithm can be refer to [20].

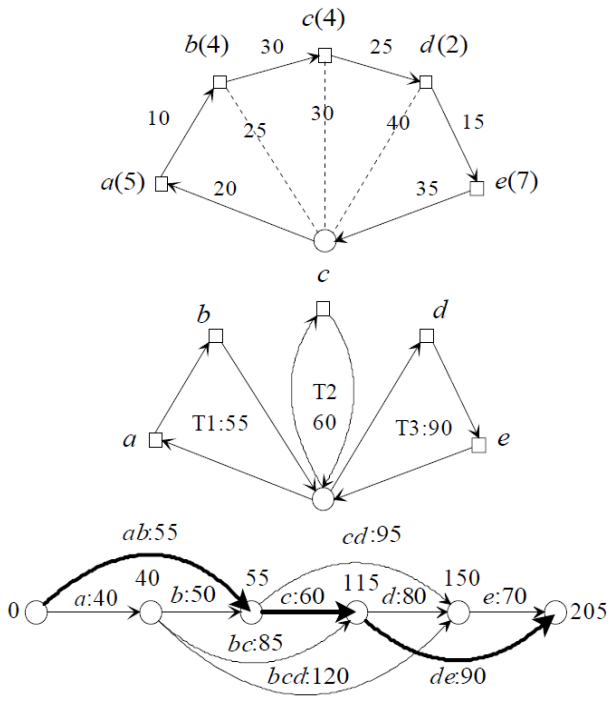

Figure 2. An example of splitting procedure by Prins[20].

2.4 Discrete PSO by Clerc[21]

In this paper, we use the Clerc method to solve our vrp problem. Let us consider a position like $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip}), i = 1, 2, \ldots, N$.

(1) Velocity is defined as an operator v which, when applied to a position during one time step, gives another position. So, here, it is a permutation of N elements, that is to say a list of transpositions.The length of this list is $|v|$. A velocity is then defined by $v = ((i_k, j_k)), k \uparrow_1^{|v|}$ which means *exchange numbers* (i_1, j_1), then numbers (i_2, j_2), etc. and at last numbers $(i_{|v|}, j_{|v|})$. A null velocity is a velocity equivalent to $\varnothing$, the empty list.

(2) Opposite of a velocity is defined by

$$\neg v = ((i_k, j_k)), k \downarrow_1^{|v|} = ((i_{|v|-k+1}, j_{|v|-k+1})), k \uparrow_1^{|v|}.$$

It means *to do the same transpositions as in v, but in reverse order* .

(3) Move (addition): *position plus velocity*

Let x be a position and v a velocity. The position $x'=x+v$ is found by applying the first transposition of v to p, then the second one to the result etc.

(4) Substraction: *position minus position*

Let x_1 and x_2 be two positions. The difference x_2-x_1 is defined as the velocity v, found by the algorithm as follow **Figure 3**, so that applying v to x_1 gives x_2 .

Step1: *i=0;*
Step2: if *i<L*, go to step4 if ith elements of x_1 equal to that of x_2;
Step3:if unequal, make the two element as a exchange number and take a 2-opt to the element of x_2;
Step4: *i=i+1*,go to step2.

Figure 3. Algorithm flow of Substraction position minus position.

For example, $x_1 = (3,2,4,1,5), x_2 = (2,4,1,5,3)$, we calculate the operator Substraction *position minus position* as follow **Figure 4**.

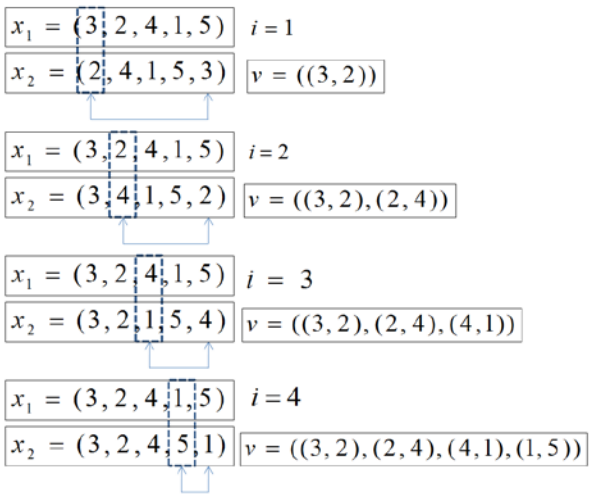

Figure 4. Illustrated example for Substraction position minus position

(5) Addition: *velocity plus velocity*

Let v_1 and v_2 be two velocities. In order to compute $v_1 \oplus v_2$ we consider the list of transpositions which contains first the ones of v_1, followed by the ones of v_2 . Optionnaly, we "contract" it to obtain a smaller equivalent velocity. In particular, this operation is defined so that $v \oplus \neg v = \varnothing$.

(6) Multiplication : *coefficient times velocity*

Let c be a real coefficient and v be a velocity. There are different cases, depending on the value of c.

(i) Case $c = 0$

We have $cv = \varnothing$.

(ii) Case $c \in (0,1)$

We just "truncate" v. Let cv be the greatest integer smaller than or equal to $c|v|$. So we define

$$cv = ((i_k, j_k)), k \uparrow_1^{c\|v\|}$$

(iii) Case $c>1$

It means we have $c=k+c'$, $k \in N^*, c' \in (0,1)$. So we define

$$cv = \underbrace{v \oplus v \oplus \cdots \oplus v}_{k \text{ times}} \oplus c'v$$

(iv) Case $c < 0$

By writing $cv=(-c)\neg v$, we just have to consider one of the previous cases.

Thus we can now rewrite Equ.1 as follow

$$\begin{cases} v_{ij}(t+1) = wv_{ij}(t) \oplus c_1 rand_1(pbest_{ij} - x_{ij}(t)) \oplus c_2 rand_2(gbest_j - x_{ij}(t)) \\ x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \end{cases}$$

1.Genate an initial population of stucture solutions;
For each particle *i=1*, ..., *S* do:
1)Initialize the particle's position with a randperm (the command can get the tour we wanted).
2)Evaluate fitness of each individual in the population;
3)Initialize the particle's best known position to its initial position: $p_i \leftarrow x_i$
4)If $(f(p_i) < f(g))$ update the swarm's best known position: $g \leftarrow p_i$
2.Until a termination criterion is met (e.g. number of iterations performed, or a solution with adequate objective function value is found), repeat:
For each particle i = 1, ..., S do:
For each dimension d = 1, ..., n do:
Pick random numbers: rand1, rand2 ~ U(0,1)
Update the particle's velocity
Update the particle's position
If (f(xi) < f(pi)) do:
Update the particle's best known position
If (f(pi) < f(g))
Update the swarm's best known position
If mod(i,20)=0
Restart the velocities of the pariticles
Now g holds the best found solution.
Until stopping criterion is satisfied

Figure 5. Scheme of the propose PSO algorithm.

From the definition of *velocity plus velocity*, we can find out the list of transpositions will be huge when after many step updates. This will make the PSO computation slowly. To conquer this difficulty, we retart the computation of

PSO based on the history information. Thus, the scheme of our PSO can be summarised as follow Figure 4.

3 Results

The whole algorithmic approach was implemented in Matlab R2013a. The parameters of the proposed algorithm are selected after thorough testing. A number of different alternative values were tested and the ones selected are those that gave the best computational results concerning both the quality of the solution and the computational time needed to achieve this solution. The algorithm was tested on two parts. The values of *c1* and *c2* are, respectively, *c1=1.4* and *c2 =1.4* and the value of w is equal to 1. The velocities are initialized with zeros and the velocities are reinitialized with zeros after 20 iterations so that computation is accelerated. The algorithm was tested on a small instance and a set of benchmark instances.

3.1 Test on a small instance

One distribution center has some vehicles. It must service for 8 customers one day. Number *0* represents the depot and numbers *1,2,...,8* represent customers. Each vehicle's capacity is *8*. We assume that all vehicles start from distribution center at *0* and must back to the distribution center. The demand of each customer, service time are described in **table 1**. The speed of each vehicle is *50* km/h. The problem is how to arrange the vehicles and its routes so as to minimize the total costs?

Table 1. This is the table caption for an example table.

ID	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
0	*0*	*40*	*60*	*75*	*90*	*200*	*100*	*160*	*80*
1	*40*	*0*	*65*	*40*	*100*	*50*	*75*	*110*	*100*
2	*60*	*65*	*0*	*75*	*100*	*100*	*75*	*75*	*75*
3	*75*	*40*	*75*	*0*	*100*	*50*	*90*	*90*	*150*
4	*90*	*100*	*100*	*100*	*0*	*100*	*75*	*75*	*100*
5	*200*	*50*	*100*	*50*	*100*	*0*	*70*	*90*	*75*
6	*100*	*75*	*75*	*90*	*75*	*70*	*0*	*70*	*100*
7	*160*	*110*	*75*	*90*	*75*	*90*	*70*	*0*	*100*
8	*80*	*100*	*75*	*150*	*100*	*75*	*100*	*100*	*0*
d_i	*0*	*1*	*2*	*1*	*2*	*1*	*4*	*2*	*2*

By conducting the proposed PSO algorithm with initial population size 50 and maximal generation 100, the optimal solution is assign *2* vehicles and their routes are 0-*2-8-5-3-1-0* and *0-6-7-4-0*. The optimal cost is *67.5*. The first vehicle's practical capacity of route *0-2-8-5-3-1-0* is *7*. The second vehicle's practical capacity of route 0-6-7-4-0 is 8. In order to verify the efficient of the proposed PSO algorithm, we also use Lingo software to find the global optimal solution of this problem. The results show that the proposed PSO algorithm can find the same optimal solution as that found by Lingo software, but the running time of the proposed PSO algorithm is much less than that of Lingo software.

3.2 Test on some benchmark instances

These benchmark instances have, initially, been proposed and used for the Capacitated Vehicle Routing Problem, but due to the fact that every variant of the Vehicle Routing Problem is a generalization of the Capacitated Vehicle Routing Problem, these benchmark instances have, also, been used in other variants of the Vehicle Routing Problem. Each instance of the set contains between 16 and 65 nodes including the depot. The locations of the nodes are defined by their Cartesian co-ordinates and the travel cost from node *i to j* is assumed to be the respective Euclidean distance. Each instance includes capacity constraints without maximum route length restrictions and with zero service time. In Table 2, the characteristics of each instance and the results of the proposed algorithm are presented. Here we set the initial population size to 200 and maximal generation 500, and each instance run 20 times. We collected the best result within these 20 times computations and listed in last column. From the results we can find that our results had big gaps with the optimal solutions. These gaps can be reduced by taking larger population size and larger number of iterations, which can be bound to employ huge computations.

Table 2. Benchmark instances proposed by Christofides [22] and results comparisons.

Instance	*n*	*Capacity*	*best*	*ours*
A-n32-k5	*32*	*100*	*784*	*974*
A-n55-k9	*55*	*100*	*1073*	*1423*
A-n60-k9	*60*	*100*	*1354*	*1762*
P-n16-k8	*16*	*35*	*450*	*603*
P-n22-k2	*22*	*160*	*216*	*284*
P-n23-k8	*23*	*40*	*529*	*734*
P-n60-k10	*65*	*120*	*744*	*953*

One of the solution traces of the instance A-n32-k5 is shown by Figure 6 which can delegate general properties of the propsed PSO on many benchmark instances. We can find that during the computation, the speed of finding better solutions became slowly as iteration times goes away. This bad convergence means that the mechamism of searching optimal solution should be improved.

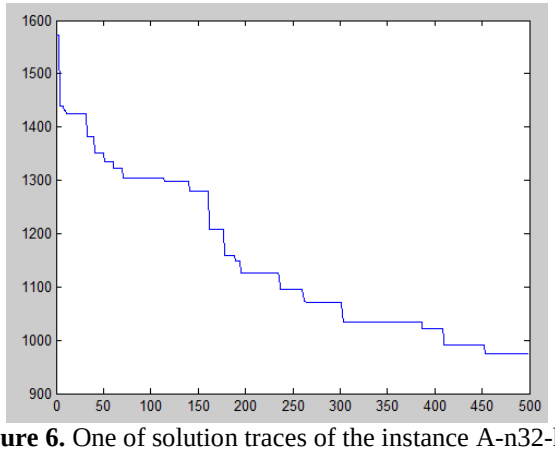

Figure 6. One of solution traces of the instance A-n32-k5.

4 Conclusion

There are many practical problems that can be represented by VRP. Since VRP is an NP-hard problem, it is difficult to find the exact solution. It is an important issue to design

effectively algorithms for solving the large size VRP problem. This paper we combine discrete PSO on tsp by clerc [21] and optimal splitting procedure found by Prins [20] to solve VRP. Experiment results show that the proposed algorithm is an efficient algorithm in solving VRP. This paper shows that the proposed algorithm algorithm can find the feasible solution effectively and almost can find the global optimal solution for small instance. For larger instances, if the size of populations in the proposed algorithm increases, the possibility of finding the global optimal solution will increase. Since the speed of finding better solutions became slowly as iteration times goes away for larger instances, our future research will be focused on how to improve and to design the mechamism of searching optimal solution.We also hope the proposed PSO algorithm can be used to solve VRP with time window.

Acknowledgments

This work was supported by the Funding Project for Academic Human Resources Development in Institutions of Higher Learning Under the Jurisdiction of Beijing Municipality (CIT&TCD20130327), and Major Research Project of Beijing Wuzi University.

References

[1] Bodin, L., Golden, B., "Classification in vehicle routing and scheduling", Networks, 1981, 11, pp. 97-108.

[2] Bodin, L., Golden, B., Assad, A., Ball, M.， " The state of the art in the routing and scheduling of vehicles and crews", Computers and Operations Research ,1983, 10, pp. 63-212.

[3] Fisher, M.L., "Vehicle routing", in: M.O. Ball, T.L. Magnanti, C.L. Momma, G.L. Nemhauser (Eds.), in: Network Routing, Handbooks in Operations Research and Management Science, North Holland, Amsterdam, 1995, 8, pp. 1-33.

[4] Gendreau, M., Laporte, G., Potvin, J.Y., "Vehicle routing: modern heuristics", in: E.H.L. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley, Chichester, 1997, pp. 311-336.

[5] Gendreau, M., Laporte, G., Potvin, J.Y., "Metaheuristics for the capacitated VRP", in: P. Toth, D. Vigo (Eds.), The Vehicle Routing Problem, Monographs on Discrete Mathematics and Applications, Siam, 2002, pp. 129-154.

[6] Laporte, G., Gendreau, M., Potvin, J.Y., Semet, F., "Classical and modern heuristics for the vehicle routing problem", International Transactions on Operations Research ,2000,7, pp. 285-300.

[7] Laporte, G., Semet, F., "Classical heuristics for the capacitated VRP", in: P. Toth, D. Vigo (Eds.), The Vehicle Routing Problem, Monographs on Discrete Mathematics and Applications, Siam, 2002, pp. 109-128.

[8] Marinakis, Y., Migdalas, A., "Heuristic solutions of vehicle routing problems in supply chain management", in: P.M. Pardalos, A. Migdalas, R. Burkard (Eds.), Combinatorial and Global Optimization, World Scientific Publishing Co, 2002, pp. 205-236.

[9] Golden, B.L., Assad, A.A., Vehicle Routing: Methods and Studies, North Holland, Amsterdam, 1988.

[10] Golden, B.L., Raghavan, S., Wasil, E., "The Vehicle Routing Problem: Latest Advances and New Challenges", Springer LLC, 2008.

[11] Pereira, F.B., Tavares, J., "Bio-inspired Algorithms for the Vehicle Routing Problem, Studies in Computational Intelligence", vol. 161, Springer, Berlin Heidelberg, 2008.

[12] Kennedy, J., Eberhart, R., "Particle swarm optimization", in: Proceedings of 1995 IEEE International Conference on Neural Networks, 1995, 4,pp. 1942-1948.

[13] Kennedy, J. , Eberhart, R., "A discrete binary version of the particle swarm algorithm", in: Proceedings of 1997 IEEE International Conference on Systems, Man, and Cybernetics, 1997, 5, pp. 4104-4108.

[14] Kennedy, J., Eberhart, R., Shi, Y., "Swarm Intelligence", Morgan Kaufmann Publisher, San Francisco, 2001.

[15] Shi, Y., Eberhart, R., "A modified particle swarm optimizer", in: Proceedings of 1998 IEEE World Congress on Computational Intelligence, 1998, pp. 69-73.

[16] Banks, A., Vincent, J., Anyakoha, C., "A review of particle swarm optimization". Part I: Background and development, Natural Computing, 2007,6(4), pp.467-484.

[17] Banks, A., Vincent, J., Anyakoha, C., "A review of particle swarm optimization". Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications, Natural Computing, 2008, 7, pp. 109-124.

[18] Clerc, M., "Particle Swarm Optimization", ISTE Ltd, London, 2006.

[19] Poli, R., Kennedy, J., Blackwell, T., "Particle swarm optimization. An overview", Swarm Intelligence, 2007, 1, pp. 33-57.

[20] Prins, C., "A simple and effective evolutionary algorithm for the vehicle routing problem", Computer and Operations Research, 2004, 31, pp. 1985-2002.

[21] Clerc, M., "Particle Swarm Optimization", New Optimization Techniques in Enginerring, 2004, pp. 219-239

[22] Christiansen, C.H., Lysgaard, J. , "A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands", Operations Research Letters, 2007, 35, pp.773-781.