Vehicle Routing Problem with Uncertain Demands: An Advanced Particle Swarm Algorithm

Abstract

The Vehicle Routing Problem (VRP) has been thoroughly studied in the last decades. However, the main focus has been on the deterministic version where customer demands are fixed and known in advance. Uncertainty in demand has not received enough consideration. When demands are uncertain, several problems arise in the VRP. For example, there might be unmet customers' demands, which eventually lead to profit loss. A reliable plan and set of routes, after solving the VRP, can significantly reduce the unmet demand costs, helping in obtaining customer satisfaction. This paper investigates a variant of an uncertain VRP in which the customers' demands are supposed to be uncertain with unknown distributions. An advanced Particle Swarm Optimization (PSO) algorithm has been proposed to solve such a VRP. A novel decoding scheme has also been developed to increase the PSO efficiency. Comprehensive computational experiments, along with comparisons with other existing algorithms, have been provided to validate the proposed algorithms.

Keywords: Vehicle routing problem, particle swarm optimization, uncertain demand.

1. Introduction

Significant efforts have been made to solve realistic problems in supply chain management and logistics (Clark and Scarf, 1960; Graves et al., 1993, among many others). The complexity of the resulting mathematical formulations is of primary concern in the real world supply chains problems. In fact, the formulations are too large and the number of binary variables goes over several hundreds or even thousands for small case studies. Therefore, in most cases, the scientific community is unable to find optimal solutions in a reasonable amount of time. Furthermore, the mathematical optimum is of no great concern because it depends on the input data that, most of the time, is merely approximated. This motivates the search for near optimal solutions by means of heuristic approaches. Advanced metaheuristic methods such as Genetic Algorithms, Ant Colony Optimization, Neural Networks, Particle Swarm Optimization (PSO) and many others have been proposed. Some authors like Yang et al., (2004) have stated the properties of PSO which defend ease of implementation and tuning of parameters. Some other researchers (Tao et al., 2008; Chen et al., 2006) claim that PSO can find solutions with relatively better qualities when hybridized with local search. Kennedy and Eberhart (1995) are believed to be the pioneers of the PSO concept which is a kind of swarm intelligent algorithm based on socio-psychological principles. It has been applied to several routing problems with success in other occasions. For example, Ai and Kachitvichyanukul (2009a) developed a PSO for a VRP with simultaneous pick-up and delivery, and compared the performance of their method with other existing metaheuristics using some benchmark problems. They used a similar PSO for the capacitated VRP (CVRP) and reported some promising results (Ai and Kachitvichyanukul, 2009b). PSO has also been applied to other logistics problems (Ai and Kachitvichyanukul, 2008; Ai and Kachitvichyanukul, 2009c). Önüt et al. (2008), for instance, used PSO for a multiple-level warehouse layout design problem. Shi et al. (2007) successfully applied PSO for the Traveling Salesman Problem (TSP).

One of the main assumptions in the general VRP is that all the input parameters and data are assumed to be deterministic (Bertsimas, 1992). Therefore, a small perturbation on the input data could result in some impractical and/or suboptimal solutions. Stochastic VRP (SVRP) was first studied by Tillman (1969) who presented a savings approach for the multi-depot SVRP. Jaillet and Odoni (1988) discussed some heuristics used to solve the probabilistic VRP. Dror (1993) modeled SVRP by a Markov Decision Process. Golden and Yee (1979) introduced a chance constrained programming model for VRPs with stochastic demands and obtained some analytic results. Stewart and Golden (1983) extended the work of Golden and Yee (1979) and presented some computational results. Gendreau et al. (1995) used an exact algorithm to solve the SVRP. Later, Gendreau et al. (1996a) applied Tabu search as a metaheuristic for the SVRP. Sungur et al. (2008) used an exact algorithm to achieve robust solutions for the SVRP. Shen et al. (2009) surveyed some real large scale cases in medical supply chains with uncertain demands or tight deadlines.

The main contributions of this paper are presenting a new decoding algorithm and applying PSO to solve the CVRP. We have also studied a special form of CVRP where demand is not certain. Furthermore, we have assumed that the demand distribution is unknown. Since solving the presented problem to optimality is not easy, we have used a variant of PSO to determine near optimal solutions. In order to demonstrate the effectiveness of our proposed PSO, results have been compared with those of Ai and Kachitvichyanukul (2009b) which we consider a reference since these authors also proposed a PSO for similar problems.

The paper has been organized as follows: introduction in section 1; problem definition in section 2; the proposed PSO algorithm for the regular deterministic CVRP, along with the decoding procedure (M1 algorithm) and a comparison with other algorithms, in section 3; uncertainty (definition, development of a method and comparison of performance) in section 4; and finally, conclusions in section 5.

2. Problem definition

VRP is a combinatorial optimization problem intended to serve a number of customers with a fleet of vehicles. Formulated initially by Dantzig and Ramser (1959), the VRP is a serious problem in the fields of transportation, distribution and logistics. In a typical VRP we have a depot, where different vehicles are to deliver goods to various customers and the primary objective is to minimize the total transportation cost or distance.

Laporte (1992) defines the classical VRP as follows: Let $G = (V, A)$ be a graph where $V = \{1, \ldots, n\}$ is a set of vertexes representing clients ($V(1)$ is the single depot) with known and deterministic demands d_i and A is the set of arcs that interconnect all clients. The cost or weight of each arc between nodes i and j is denoted by c_{ij}. In addition, there are m vehicles available. The objective is to minimize the total travelled distance by these vehicles, subject to:

(i) each client in $V \backslash \{1\}$ is visited only once;

(ii) each vehicle route starts and ends at the depot;

Considering uncertainty in the VRP results in more real life-like problems which should be solved in such a way that the solutions would be robust against perturbations caused by the uncertainty. The findings in the robust optimization field are noticeable (Ben-Tal and Nemirovski, 1998; El Ghaoui et al., 1998; Ben-Tal and Nemirovski, 1999). El Ghaoui (2003) illustrates the robust solution by a conceptual example which is depicted in Figure 1. The bold quadrangle is the feasible area for a typical problem with deterministic parameters. When the parameters are perturbed, the intersection of feasible areas is the robust area shown by the black boundary. Since it is impossible to model this area analytically, a counterpart is replaced as the robust feasible area. This way, all the solutions to the existing models in this counterpart are robust, but their best solutions are a little worse than the robust optimum solution. So in the robust optimization, there is a trade-off between robustness and quality of the solution. Despite quality degradation, in some cases it is priceless to get a robust solution. Ben-Tal and Nemirovski (1999) proposed two methods which formulate a mathematical robust counterpart.

Figure 1. Conceptual representation of the robust optimization.

Since the VRP is an NP-Hard problem (Lenstra and Rinooy Kan, 1981), there have been tremendous efforts made to use metaheuristics to find near optimal solutions for it. One for example, is the PSO which has attracted a lot of attention in many different types of optimization problems (Ai and Kachitvichyanukul, 2009a; Poli, 2008). We have chosen the PSO because of some advantages such as easy implementation and rapid convergence, as cited by several authors (Angeline, 1998; Yang et al., 2004). Furthermore, PSO has not yet been explored to the SVRP, and still some interesting results are to be discovered. Since PSO solves problems with continuous variables and the VRP is a combinatorial optimization problem, a decoding method is needed to apply PSO to solve the VRP. Therefore, designing an effective decoding algorithm can significantly improve the solutions given by the PSO.

3. Proposed PSO algorithm

PSO uses some multidimensional particles, indicating position and velocity, to model a swarm. Each particle moves through space (i.e., in $\Re^n$) while updating its own best position, the global best position and its neighborhood's best position. It informs other particles about its best position, it also obtains theirs and then adjusts its own position and velocity according to the shared information. Figure 2 illustrates how a particle velocity is calculated. Consider some two-dimensional particles. One of them (in solid black with coordinates (2, 3)) has been highlighted to show how a particle position is updated. Three vectors are considered for this particle as follows:

- A global best position (shown in green with coordinates (9, 7)) is updated when a new best position is found by the particles in the swarm.
- A neighborhood best position (shown in yellow with coordinates (2, 6)) which is the best in the neighborhood of each particle.
- A local best position which is the best that the particle has experienced so far (coordinates (1, 4)).

Figure 2. Particle position and velocity.

The red vector, representing velocity, is the resultant of these three vectors plus the last velocity of the black particle (coordinates (0.1, -0.1)). In order not to loose some potential solutions at each iteration, the velocity vector should be so calculated as to be small enough to avoid large perturbations in the solutions. This is why the coefficients are selected as uniform random variables. In our example, these coefficients are 0.1, 0.3 and 0.2 and have been used as follows:

New velocity = $\begin{bmatrix} 0.1 \\ -0.1 \end{bmatrix} + 0.1(\begin{bmatrix} 9 \\ 7 \end{bmatrix} - \begin{bmatrix} 2 \\ 3 \end{bmatrix}) + 0.3(\begin{bmatrix} 2 \\ 6 \end{bmatrix} - \begin{bmatrix} 2 \\ 3 \end{bmatrix}) + 0.2(\begin{bmatrix} 1 \\ 4 \end{bmatrix} - \begin{bmatrix} 2 \\ 3 \end{bmatrix}) = \begin{bmatrix} 0.6 \\ 1.4 \end{bmatrix}$

Based on the above example, the new position of the black particle (shown in gray) will be defined by the following resultant vector:

New position = $\begin{bmatrix} 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 0.6 \\ 1.4 \end{bmatrix} = \begin{bmatrix} 2.6 \\ 4.4 \end{bmatrix}$

All particles should be updated following the previous example. Afterwards, the global best, the neighborhood best and the local best positions are updated.

In this simple example, we have used two-dimensional particles, but depending on the problem, any suitable number of dimensions can be selected as the particle string length.

The notations related to our proposed PSO algorithm are as follows:

α	iteration index; $\alpha = 1,2,\dots,T$
k	particle index, $k = 1,2,\dots,K$
s	dimension index, $s = 1,2,\dots,S$
u	uniform random number in the interval [0, 1]
$w(\alpha)$	inertia weight in the α^{th} iteration
$v_{ks}(\alpha)$	velocity of the k^{th} particle at the s^{th} dimension in the α^{th} iteration
$\theta_{ks}(\alpha)$	position of the k^{th} particle at the s^{th} dimension in the α^{th} iteration
η_{ks}	personal best solution (pbest) of the k^{th} particle at the s^{th} dimension
η_{gs}	global best solution (gbest) at the s^{th} dimension
η_{ks}^{K}	local best solution (lbest) of the k^{th} particle at the s^{th} dimension
η_{ks}^{N}	near neighbor best solution (nbest) of the k^{th} particle at the s^{th} dimension
C_p	personal best solution acceleration constant
C_g	global best solution acceleration constant
C_k	local best solution acceleration constant
C_n	near neighbor best solution acceleration constant
$\theta_{\max}$	maximum position value
$\theta_{\min}$	minimum position value

Θ_k	position vector of the k^{th} particle
Ω_k	velocity vector of the k^{th} particle, $[\omega_{k1}, \omega_{k2}, ..., \omega_{kS}]$
η_k	personal best solution vector of the k^{th} particle
η_g	global best solution vector
η_k^K	local best solution vector of the k^{th} particle
$Z(x)$	objective value of x
FDR	fitness distance ratio

The proposed PSO algorithm, based on the above notations, may now be defined as follows:

1. Initialize K particles as a swarm, generate the k^{th} particle with a random position Θ_k in the range [$\theta_{\min}, \theta_{\max}$], velocity Ω_k = 0 and personal best $\eta_k = \Theta_k$ for $k = 1,2,\ldots,K$. Set iteration $\alpha = 1$.

2. For $k = 1,2,\ldots,K$, calculate the objective value of Θ_k $(Z(\Theta_k))$ according to the proposed decoding M1 algorithm (to be explained later).

3. Update pbest: if $Z(\Theta_k) < Z(\eta_k) \Rightarrow \eta_k = \Theta_k$; $k = 1,2,\ldots,K$

4. Update gbest: if $Z(\eta_k) < Z(\eta_g) \Rightarrow \eta_g = \eta_k$; $k = 1,2,\ldots,K$

5. Update lbest: among all the k^{th} particle's neighboring pbests, set the one with the least objective value in η_k^K. $k = 1,2,\ldots,K$

6. Generate nbest: set $\eta_{ks}^N = \eta_{os}$ (η_{os} is the same as η_{ks} : personal best solution of the o^{th} particle at the s^{th} dimension; $o = 1,2,\ldots,K$ and $k \neq o$) ; $k = 1,2,\ldots,K$; $s = 1,2,\ldots,S$

to maximize the following Fitness Distance Ratio (FDR). In other words, the s^{th} dimension of the k^{th} panicle's velocity is updated using a particle called the nbest, with a prior best position η_{ks}^N, chosen to maximize.

$$FDR = \frac{Z(\Theta_k) - Z(\eta_o)}{|\theta_{ks} - \eta_{os}|}; \quad k \neq o \text{ and } o = 1,2,\ldots,K \qquad (1)$$

Some researchers (Ozcan and Mohan, 1999; Clerc and Kennedy, 2002) have shown that particles move in sinusoidal waves until they reach the global best positions detected by all particles so far. If a point, visited by a particle during this oscillation, has a better objective value than its previous best position, then particle movement continues, generally converging at the global best position discovered so far. Other particles behave similarly, converging at a good local optimum for the problem. However, if the global optimum for the problem doesn't lie on a path between the original particle position and such a local optimum, then the global optimum is not achieved and the particles are wasting computational effort in seeking to move in the same direction (towards the local optimum), whereas better results may be obtained if various particles explore other possible search directions. Peram et al. (2003) has introduced another alternative in which the particles are influenced by other particles, not just move towards or away from the best position found so far. The socio-cognitive learning process, defined in the standard PSO, is based on a particle's own experience and that of the most successful particle. The FDR adds a new dimension to this approach; each particle also learns from the experience of the neighboring particles that have a better fitness than itself. FDR computes a best neighborhood position for each particle by maximizing the ratio between the objective difference of each particle for each dimension and the absolute value of the difference between the particles positions in that dimension (Veeramachaneni et al., 2003). Ai and Kachitvichyanukul (2009b) too have shown that FDR affects VRP.

7. Update velocity and position of the k^{th} particle:

$$w(\alpha) = w(T) + \frac{\alpha - T}{1 - T}\left[w(1) - w(T)\right] \tag{2}$$

where:

$w(1)$ and $w(T)$ are the input parameters ($w(1) > w(T)$).

$w(\alpha)$ is a parameter that decreases as the number of iterations increases.

$$\begin{aligned} v_{ks}(\alpha+1) = w(\alpha)v_{ks}(\alpha) + c_p u(\eta_{ps} - \theta_{ks}(\alpha)) + c_g u(\eta_{gs} - \theta_{ks}(\alpha)) \\ + c_k u(\eta_{ks}^K - \theta_{ks}(\alpha)) + c_n u(\eta_{ks}^N - \theta_{ks}(\alpha)) \end{aligned} \tag{3}$$

In Equation (3), velocity of the k^{th} particle is therefore based on the previous iteration velocity, local best position, global best position, neighborhood best position and FDR.

$$\theta_{ks}(\alpha+1) = \theta_{ks}(\alpha) + v_{ks}(\alpha+1) \tag{4}$$

After determining the position of the k^{th} particle, according to Equation (4), if it is less than $\theta^{\min}$ or more than $\theta^{\max}$, it should be considered equal to $\theta^{\min}$ or $\theta^{\max}$ respectively and then the velocity is set to zero.

8. If stop criterion is met ($\alpha = T$), stop. Otherwise, $\alpha = \alpha + 1$ and return to step 2.

3.1. M1 decoding algorithm

In the PSO algorithm proposed in this research, every particle is represented by an array of real numbers. The particle moves in a multidimensional space to find the optimal /near optimal position.

So the important point is: How is a particle position used to determine the vehicles routes in a VRP? This study has proposed a decoding algorithm with the help of which we may define the characters of each particle. The algorithm should first specify the length of the array by showing haw many characters each particle has. Ai and Kachitvichyanukul (2009b) are the only researchers who have used such algorithms for the interpretation of such arrays. In the algorithm proposed in this research, the length of each particle equals $3n$ where n is the number of customers. The first n characters indicate the priority of the customers to be visited (similar to Ai and Kachitvichyanukul, 2009b) and the second and the third n characters are used to assign customers to vehicles. An example of assigning customers to a vehicle is as fallows:

Suppose there are 7 customers to be assigned to a vehicle. Figure 3 shows customers scores, extracted from a random particle. These scores are shown in the descending order in Figure 4.

Figure 3. Customers scores extracted from a particle.

Figure 4. Customers scores sorted in the descending order.

Some problems will arise if one uses a traditional sequencing method; e.g. route [7-4-3-5-2-1-6] will be found for the above example. Logically, Route 1 ([7-4-3-5-2-1-6]) is the same as Route 2 [6-1-2-5-3-4-7] but only backwards. There are many other redundancy problems that can severely affect the algorithm. Suppose [6-1-2-3-5-4-7] is the best route. Then, the PSO algorithm will guide Route 1 to this best route. Therefore, it tries to increase Customer 6's score (17) so that this customer is served sooner, and also tries to decrease Customer 7's score (82) because it is the last visited costumer in the best known solution and so on. The progressive updating of the scores needs several iterations. In the mean time, an updated route could be something like [4-7-5-3-6-2-1]. It is also possible that Route 1 is much better than the updated route. In order to overcome this problem, the customers are added sequentially from the leftmost odd columns and then from the rightmost even columns when determining a route. This procedure for the above example is depicted in Figure 5 where the depot is denoted as D.

Figure 5. Sequence of vehicles services to customers, based on Figure 4.

The second and the third n characters are used to assign customers to vehicles. These two strings can be applied independently. If we use only one string, the chance of remaining in local search will increase. In order to avoid local optimum convergence, one of these strings is selected stochastically. A threshold parameter, $p \in [0,1]$, is used to select one of these two strings. Therefore, the second string is applied with a probability p and the third one with a probability $1-p$. After selecting one of the 2nd or the 3rd strings, the interval between $(\theta_{\min}, \theta_{\max})$ will be stratified by Tl_t which is given by Equation (5) below. Thereafter, if the value of θ_i lies between Tl_t and Tl_{t+1} , it means that the i^{th} customer will be served by the t^{th} vehicle.

$$Tl_t = \left(\frac{\theta_{\max} - \theta_{\min}}{m} \right)(t-1) + \theta_{\min} \tag{5}$$

As an example, suppose the second string is selected (Figure 6). In order to assign 7 customers to 3 vehicles we may proceed as follows:

Figure 6. The second string of n characters from a particle.

Based on Equation 5, Tl_1, Tl_2, Tl_3 and Tl_4 are 0, 33, 66 and 99 respectively (supposing that $\theta_{\min} = 0$ and $\theta_{\max} = 99$). If we define V_t as a customer set served by the t^{th} vehicle, then: $V_1 = \{2, 7, 6\}, V_2 = \{5,1\}$ and $V_3 = \{3, 4\}$

Many researchers suggest employing a hybrid strategy which embeds a local optimizer between the iterations of the metaheuristics (Tao et al., 2008; Chen et al., 2006). So, after determining customer assignments and vehicle service sequences, and in order to improve the solution, some local search algorithms can be applied. The ones we have used have been Variable Neighborhood Search (VNS) with algorithms such as 2-OPT, Exchange1-1 and Iterated Greedy to be explained in Section 3.2. It should be noted that using a local search algorithm means decreasing the objective function value without modifying the particle dimensions. Then, if the difference between objective function values before and after local search (B_l and A_l respectively) is considerable, the related particle might be considered as a local best. This is the reason of the movement of other particles toward this one, while the real local best position is just different. Therefore, if the difference in the objective function values is greater than a predefined value δ ($0.1 * B_l$ in this research) the velocity vector, related to the particle, will be zero in that iteration. In addition, the particle is so updated that in the next iteration, before implementing the local search, the same solution (the one obtained in previous iteration after local search) will be obtained. This updating procedure is clarified with a numerical example in Step 5 of the M1 decoding algorithm. This algorithm is illustrated by the following example:

Suppose 7 customers (n=7) are to be served by 3 vehicles (m=3) (Table 1). Location of each customer is defined by its coordinates; for example, Customer 1is located at (82, 90). A 21-position-long particle (3*7) (Figure 7) has been decoded by the M1 algorithm using the following data:

$$(\theta_{\min}, \theta_{\max}) = (0,99), \quad p = 0.9, \quad (Tl_1, Tl_2, Tl_3, Tl_4) = (0,33,66,99), \quad \delta = 0.1 * B_l$$

Table 1. The input data of an example VRP

Costumer No.	(Depot)	1	2	3	4	5	6	7
Coordinates	82	82	99	70	80	58	93	62
	50	90	51	85	20	60	60	45

Figure 7. A 21-position-long particle.

1. Assign the n customers to the vehicles V_t ; $t = 1,2,\ldots,m$

1.1. In order to assign the h^{th} customer of the i^{th} particle, if $rand$ (a random variable value generated from the interval $[0,1]$) is less than p , then $\theta_{i(n+h)}$ and otherwise, $\theta_{i(2n+h)}$ is selected.

1.2. For the selected θ in the previous stage, a vehicle that satisfies equation $Tl_t \le \theta < Tl_{t+1}$ is selected considering the capacity constraint.(If this constraint is violated, the objective function will be set to a big value as the penalty and algorithm will terminate.)

If the random variable is 0.8, we should use the second 8 characters, because $0.8 < p$.

So, according to Tl_t:

$V_1 = \{2,6,7\}, V_2 = \{1,5\}$ and $V_3 = \{3,4\}$

The numbers in the above three sets do not show the sequence of customers; this sequence will be determined in the next step.

2. Define the sequence of customers assigned to the t^{th} vehicle; $t = 1,2,3,\ldots,m$

2.1. Assume P set as follows:

$P_t = \{\theta_{i,(V_t)_z} \mid z = 1,2,...,Z\}$; $Z =$ Number of V_t members

In our example:

$P_1 = \{19,17,82\}, P_2 = \{18,24\}$ and $P_3 = \{65,81\}$

2.2. Sort P in the descending order and update the sequence of customers in V_t according to Figure 5.

$V_1 = (7-2-6), V_2 = (5-1)$ and $V_3 = (4-3)$

3. Calculate the objective function value (B_l).

$\cos t_{V_1} = 81, \cos t_{V_2} = 104, \cos t_{V_3} = 133 \Rightarrow B_l = 81 + 104 + 133 = 318$

4. Improve via local search. (Local search algorithms have been presented in Section 3.2) After improvement, suppose we have the following modified routes:

$V_1 = (7-2-6), V_2 = (3-1), V_3 = (4-5)$

5. Update the particle and its related velocity

5.1. Calculate the objective function value (A_l)

$\cos t_{V_1} = 78,\ \cos t_{V_2} = 90,\ \cos t_{V_3} = 102 \Rightarrow A_l = 78 + 90 + 102 = 270$

5.2. If $B_l - A_l \le \delta$, then consider A_l as the objective function value and exit the algorithm.

Since $318 - 270 > 31.8$, we should go to the next step.

5.3. Update the particle only for the first and the second n characters.

5.3.1. In order to update $\theta_{i,(V_t)_k}$, which belongs to V_t with K customers, generate a random value between $\theta_{\min}$ and $\theta_{\max}$ that satisfies the following:

$\cdots < \theta_{i,(V_t)_{k-2}} < \theta_{i,(V_t)_2} < \theta_{i,(V_t)_{k-1}} < \theta_{i,(V_t)_1}$.

So, for our example we will have:

For the first route: $\theta_{i,2} < \theta_{i,8} < \theta_{i,6} < \theta_{i,7} \Rightarrow \theta_{i,2} = 17, \theta_{i,6} = 63, \theta_{i,7} = 72$

For the second route: $\theta_{i,1} < \theta_{i,3} \Rightarrow \theta_{i,1} = 32, \theta_{i,3} = 58$

For the third route: $\theta_{i,5} < \theta_{i,4} \Rightarrow \theta_{i,5} = 10, \theta_{i,4} = 79$

5.3.2. Update $\theta_{i,n+(V_t)_k}$ for all customers related to the t^{th} vehicle based on:

$Tl_t < \theta_{i,n+(V_t)_k} < Tl_{t+1}$.

Since the 7th, the 2nd and the 6th customers should be served by the first vehicle, the related θ values should be generated randomly between $Tl_1 = 0$ and $Tl_2 = 33$.

$\theta_{i,(8+2)} = 31, \theta_{i,(8+6)} = 23, \theta_{i,(8+7)} = 8$

This procedure is repeated for the customers served by the second and the third vehicles:

$\theta_{i,(8+1)} = 54, \theta_{i,(8+3)} = 42$

$\theta_{i,(8+5)} = 90, \theta_{i,(8+4)} = 79$

5.4. In order to update the velocity vector (V_i), the first and the second n characters of V_i are all assumed equal to zero.

The above steps are demonstrated in Figure 8 too. The proposed decoding algorithm has three main novel features compared with the decoding methods SR-1 and SR-2 presented by Ai and Kachitvichyanukul, (2009b). First is the customer sequencing procedure (illustrated in Figure 5) which avoids some redundancy problems. Second is the special way in which the customers are assigned to vehicles avoiding convergence at a local optimum. The key point here is the usage of two different parallel strings. According to the second step of M1, often the first, and sometimes the second, string is used to assign customers to vehicles which helps escaping a local optimum. And the last is the updating of the particles values when some relative improvements occur after applying the local search algorithms. These algorithms may cause some route modifications that may not be reflected in the particles values. Imagine a particle with a given objective function value of 1000. If, after using the local search algorithms, the objective function improves to 900 and this particle becomes the global best, then other particles will approach this one to improve theirs. But actually, 900 is an unreal and a potential value for that particle and it will cause confusion in the particles velocity vectors because it is not reflected in the particle’s real vectors.

Figure 8. Example of the M1 decoding algorithm.

3.2. Local search algorithms

After applying the decoding algorithm, some local search algorithms are employed to increase convergence speed to better solutions. Some of such algorithms which can be applied for the VRP can be seen in Cordeau et al. (2005), Laporte, et al. (2000) and Kindervater and Savelsbergh (1997). In this research, the Variable Neighborhood Search (VNS) method is used with other algorithms such as 2-OPT, Exchange1-1 and Iterated Greedy (IG) which are defined below.

The VNS is a well-known heuristic search method applied successfully to some VRPs (Hansen and Mladenovic, 2003). For more details see Polacek et al., (2004) or Pirkwieser and Raidl, (2008). IG is a modern interesting simple heuristic method that generates solutions in two steps: stochastic destruction and construction (Ruiz and Stützle, 2007; Ruiz and Stützle, 2008). These algorithms are well-known; however, for the sake of reproducibility, they are detailed in the following subsections:

3.2.1. OPT algorithm

OPT is a well known local search algorithm with a $O(n^2)$ computational complexity that can improve VRP solutions by changing the sequence of customers for each vehicle or route. The pseudo code related to the algorithm is as follows:

```
for $i = 1$ to $(K-2)$
    for $j = (i+2)$ to $K$
        $V_{t_{(i+1)}} \leftrightarrow V_{t_j}$
        If the total cost is improves, modify the sequence $V_t$.
    end
end
```

3.2.2. Exchange 1-1 algorithm

In this algorithm, exchanges of customers among vehicles or routes are possible, so all the exchanges will be taken into account. If the distance between two customers is less than ε, the two customers will be exchanged and the objective function will be updated. If the solution is feasible and improves the objective function, the customers assignments to the vehicles are updated and other assignments are checked; otherwise, the algorithm will proceed without updates. The computational complexity of this algorithm is $O(n^2)$ in the worst case. Figure 9 and the following pseudo code illustrate the exchange procedure for two routes:

```
for $i = 1$ to $n1$ (number of customers in the 1st route)
    for $j = 1$ to $n2$ (number of customers in the 2nd route)
        if distance($Cus_i$, $Cus_j$)< $\varepsilon$
            exchange $Cus_i$ with $Cus_j$
            if solution improve, modify the routes according to this exchange
        end
    end
end
```

Figure 9. Exchange 1-1 local search.

3.2.3. Iterated Greedy (IG) algorithm

In this algorithm, r customers are selected randomly and removed from their respective routes. Then, each of the removed customers is greedily added to the remaining incomplete routes in the cheapest possible way.

These steps are shown in Figure 10. Let's suppose ten customers around a depot which are served by two vehicles. First, four customers are selected randomly. Then they are removed from the routes. Finally, they are placed in such an order that they improve the objective function. Now, two of the formerly selected customers are placed in their previous route and the other two are placed in new routes. The computational complexity of this algorithm is $O(n^2 r)$ .The details are as follows:

1. Select r customers randomly and remove them from their related vehicles.
2. Suppose R is a set including the r removed customers sorted in the descending order according to their demands.
3. Suppose V is a set of vehicles sorted in the descending order of their remaining capacities.
4. Substitute the i^{th} customer of R in the j^{th} member of V. If the solution is feasible, update V.
5. If $j < m$, substitute $j+1 \rightarrow j$ and go to step 4; otherwise, substitute $i-1 \rightarrow i$ and $j+1 \rightarrow j$ and go to step 4.

6. Find the objective function value and update the vehicles capacities.
7. If r is empty, the algorithm is finished; otherwise, go to Step 4.

Figure 10. An example of the Iterated Greedy steps.

3.2.4. VNS algorithm

Each of the three previous local search methods induces a neighborhood. The VNS searches each neighborhood until local optimality is achieved. If, at any step, the solution is improved, the search will start from the first neighborhood. Therefore, VNS is finished if and only if the solution is a local optimum with respect to all neighborhoods:

```
while solution improves
    while solution improves
            while solution improves
                    apply the 2-OPT algorithm
            end
            apply exchange 1-1 algorithm
    end
    for $n$=1 to 5
        apply IG algorithm
    end
end
```

Applying PSO decoding and local search algorithms simultaneously results in high quality solutions. The effectiveness of the M1 decoding method, in comparison with SR-1 and SR-2 algorithms (Ai and Kachitvichyanukul, 2009b), is presented in Table 2. SR-1 and SR-2 algorithms assume a virtual position for each vehicle. Each customer that has a shorter distance to a vehicle, will have more priority to be allocated to that vehicle. This approach generates some relatively symmetric tours with the vehicle centricity. The proposed algorithms are coded in the Matlab language and can be run on a personal computer with a 3.0 GHz Pentium 4 processor and 512MB RAM. The sample instances have been extracted from Augerat et al. (1995). Initial experiments indicated that our proposed PSO was rather robust as regards the working parameters. Therefore, we just employed the following parameters, mainly extracted from Chen et al. (2006) and Ai and Kachitvichyanukul (2009b), for our proposed PSO: T, number of iterations: 500; K, number of particles: 50; $w(1)$, the first inertia weight: 0.9; $w(T)$, the last inertia weight: 0.4; C_p, the personal best acceleration constant: 0.7; C_g the global best acceleration constant: 0.3; C_k the local best acceleration constant: 1.5; C_n the near neighbor best acceleration constant:1.5; $\theta_{\max}$ the maximum position value: 100; $\theta_{\min}$ the minimum position value: 0; number of iterations for the IG algorithm: 5; and number of neighborhoods: 5. In order to compare our method with SR-1 and SR-2, the following error measures have been considered:

$$SR-1_E = Objective\ function_{SR-1} - Objective\ function_{optimum}$$

$$SR-2_E = Objective\ function_{SR-2} - Objective\ function_{optimum}$$

$$M1_E = Objective\ function_{M1} - Objective\ function_{optimum}$$

Table 2. Comparison of M1 algorithm with SR1 and SR2 (Ai and Kachitvichyanukul, 2009b).

No.	Sample	objective function			
		optimum solution	SR-1	SR-2	M1
1	An32k5	784	**784**	**784**	**784**
2	An33k5	661	**661**	**661**	**661**
3	An33k6	742	**742**	**742**	**742**
4	An34k5	778	**778**	**778**	**778**
5	An36k5	799	**799**	**799**	**799**
6	An37k5	669	670	**669**	**669**
7	An37k6	949	**949**	**949**	**949**
8	An38k5	730	**730**	**730**	**730**
9	An39k5	822	825	**822**	**822**
10	An44k6	937	940	940	**937**
11	An46k7	914	**914**	**914**	**914**
12	An60k9	1354	1366	1355	**1354**
13	Bn31k5	672	**672**	**672**	**672**
14	Bn34k5	788	**788**	**788**	**788**
15	Bn35k5	955	**955**	**955**	**955**
16	Bn38k6	805	**809**	**805**	**805**
17	Bn39k5	549	**549**	**549**	**549**
18	Bn41k6	829	**829**	**829**	**829**
19	Bn43k6	742	**742**	**742**	**742**
20	Bn44k7	909	915	912	**909**
21	Bn45k5	751	**751**	**751**	**751**
22	Bn45k6	678	**678**	**678**	**678**
23	Bn50k7	741	748	746	**741**
24	Bn50k8	1312	1315	**1312**	**1312**
31	Bn64k9	861	866	863	863
32	Bn66k9	1316	1318	**1316**	**1316**
33	Bn67k10	1032	1035	1034	1035
34	Bn68k9	1272	1278	1274	**1272**
35	Bn78k10	1221	1239	1223	**1221**
36	En30k3	534	541	**534**	**534**
37	En51k5	521	**521**	**521**	**521**
38	En76k7	682	691	687	**682**
39	Fn72k4	237	**237**	**237**	**237**
40	Fn135k7	1162	1184	1165	1167
41	Mn101k10	820	821	**820**	821
42	Mn121k7	1034	1041	1036	1035
43	Pn23k8	529	**529**	**529**	**529**
44	Pn40k5	458	**458**	**458**	**458**
45	Pn45k5	510	**510**	**510**	**510**
46	Pn50k7	554	**554**	**554**	**554**
47	Pn50k8	631	**631**	**631**	**631**
48	Pn50k10	696	**696**	**696**	**696**
49	Pn51k10	741	**741**	**741**	**741**
50	Pn55k7	568	**568**	**568**	**568**
51	Pn55k8	588	**588**	**588**	**588**
52	Pn55k10	694	696	**694**	**694**
53	Pn55k15	989	998	995	**989**
54	Pn60k10	744	746	**744**	**744**

25	Bn51k7	1032	1033	**1032**	**1032**	55	Pn60k15	968	970	**968**	**968**
26	Bn52k7	747	**747**	**747**	**747**	56	Pn65k10	792	799	795	**792**
27	Bn56k7	707	709	**707**	**707**	57	Pn70k10	827	828	**827**	**827**
28	Bn57k7	1153	**1153**	**1153**	**1153**	58	Pn76k4	593	599	594	**593**
29	Bn57k9	1598	1603	**1598**	**1598**	59	Pn76k5	627	628	**627**	629
30	Bn63k10	1496	1500	1499	**1496**	60	Pn101k4	681	686	683	688

$$Error\,Averages: \quad SR-1_E = 2.77,\ SR-2_E = 0.75,\ M1_E = 0.35$$

Note: Bold values equal optimum solutions.

As we can see in Table 2, the error averages show that our proposed decoding algorithm M1 is much closer to the optimum solution in comparison with the two other decoding algorithms. According to Table 2 data, applying paired T- test for the null hypothesis $H_0 : \mu_{M1\,Error} = \mu_{SR1\,Error}$ against $H_1 : \mu_{M1\,Error} < \mu_{SR1\,Error}$, based on the Minitab reported P-value (P-value=0), will result in the rejection of H_0. So, M1 method is statistically better than SR-1. Also, applying the same test for the null hypothesis $H_0 : \mu_{M1\,Error} = \mu_{SR2\,Error}$ against $H_1 : \mu_{M1\,Error} < \mu_{SR2\,Error}$, based on the reported P-value (P-value=0.032), will result in the rejection of H_0. So, M1 method is also statistically better than SR-2. Conceptually, SR-1 and SR-2 work in such a way that they consider a position for each vehicle. The customers geographically closer to the vehicle have higher priorities to be satisfied by that vehicle. The above concept causes SR-1 and SR-2 not to be able to sweep all the possible routes and consequently they cannot find the optimum solution in many cases. These algorithms are not able to achieve the global best solution in some cases because they cannot define some asymmetric tours. In the following example, we will solve a problem with SR-1 and M1 in a similar condition (CPU time). Figures 11 and 12 are related to SR-1and M1with 823 and 789 as their objective function values respectively. As shown, M1 has found some asymmetric tours, therefore it is a better solution. Many other examples can be found as this behavior is easily reproducible.

Figure 11. A VRP which is solved by SR-1.

Figure 12. A VRP which is solved by M1.

Now, including uncertainty demand and comparing the proposed PSO and M1 with other SVRP algorithms, we will proceed.

4. Uncertainty in demands

Stochastic Vehicle Routing Problems (SVRP) have different models: VRP with Stochastic Demands (VRPSD), VRP with Stochastic Customers (VRPSC), VRP with Stochastic Customers and Demands (VRPSCD) and so on. Bertsimas (1992) presented an a priori method with diverse re-optimization policies to solve the VRPSD. Gendreau et al. (1996b) proposed some methods and techniques to solve these problems. Markovic et al. (2005) modeled a VRPSD which could be applied for delivery and pickup processes of mail, packages, and recycled materials and, concurrently, Dessouky et al. (2005) suggested a VRPSD for some health care delivery problems. Demand uncertainty is a serious problem appearing in the VRP which leads to unmet demands (Sungur et al., 2008). Novoa and Storer (2009) developed an approximate dynamic programming approach to solve the VRPSD. Therefore, it seems necessary that some methods be developed to provide solutions that are robust against uncertainty. To comprehend robustness of a solution against demand uncertainty, a conceptual example is given as follows:

Table 3. The input data of a VRP, including 7 clients served by one depot.

Costumer No.	(Depot)	1	2	3	4	5	6	7
Coordinates	82	82	99	70	80	58	93	62
	50	90	51	85	20	60	60	45
Demand	0	46	46	44	32	10	34	45
Each vehicle capacity: 100								

Figure 13 illustrates two solutions to the example problem of Table 3. The left figure depicts the optimum solution. It is clear that Route 1 is not robust against demand perturbations because summation of the demands in Route 1 is exactly equal to the related vehicle capacity, and any increase in demands will cause unmet demands leading to some profit loss. In the right figure, the cost increases about 2% (from 230 to 235), but all routes are more robust against demand uncertainty. In a typical bad case, suppose that the demands of customers 2 and 4 increase to 51 and 49 respectively (about 10% perturbation); then, the vehicle is able to satisfy the customers without any unmet demands. So, we should determine the values of 2% extra cost (related to the robust solution) and 10% unmet demands (related to the best solution) to figure out if the robust solution is more economical. In this strategy, the vehicles' remaining capacities are leveled, considering minimization of the extra cost.

Figure 13. Best (left) and robust (right) solution.

Uncertainty in demand can be modeled in different ways such as "probability distributed demands" and "Fuzzy demands" (Erbao and Mingyong , 2009; Liu and Lai, 2002). This research considers no known or assumed distribution for demands and modifies the PSO objective function in such a way that robust VRP solutions can be achieved. The modification is as follows:

Let $\tilde{d}_i$ be the customer demand ($d_i - \varepsilon d_i \leq \tilde{d}_i \leq d_i + \varepsilon d_i$) where ε is a constant indicating the perturbation percentage and d_i is the nominal demand. Therefore:

$$D_l \leq \sum_i \tilde{d}_i \leq D_u \tag{6}$$

where D_l and D_u are the lower and the upper bounds for the sum of the real perturbed demands respectively. As explained in the example of Figure 13, we should balance the unused capacity of vehicles. There, the unused capacities of vehicles for the 3 routes, when the solution is deterministic, are: 0, 20 and 23 (100- 100, 100-80 and 100- 77), but in the robust solution they change to: 10, 20 and 13. Index ψ in relation 7 below, has been devised to compare the balancing of unused capacities in different solutions.

$$\psi = (Var(\lambda_v))^{-1} \tag{7}$$

where $\lambda_v = \rho_v - \varphi$ is the difference between the unused capacity (ρ_v) for different routes (v), and $\varphi = \max\left({\sum_v \rho_v}/{n}, {(D_u - \sum_v d_v)}/{n} \right)$ is the maximum average demand considering uncertainty in demands.

ψ increases when all routes are leveled and have enough unused capacities to overcome uncertainty. Equation (7) is to make sure that each vehicle's remaining capacity limit does not exceed its upper bound.

To make Equation (3) (for the deterministic case) usable for the uncertain conditions, index ψ is added and Equation (8) is obtained as follows:

$$\begin{aligned} v_{ks}(\alpha+1) = w(\alpha)v_{ks}(\alpha) + c_p u(\eta_{ps} - \theta_{ks}(\alpha))(\psi_p) + c_g u(\eta_{gs} - \theta_{ks}(\alpha))(\psi_g) \\ + c_k u(\eta_{ks}^K - \theta_{ks}(\alpha))(\psi_K) + c_n u(\eta_{ks}^N - \theta_{ks}(\alpha))(\psi_N) \end{aligned} \tag{8}$$

Suppose in a deterministic solution the *lbest* has a high ψ and the *gbest* has a low ψ, therefore the velocity vector will tend to the *lbest* rather than to the *gbest*. Generally speaking, velocity vectors are more affected by solutions which have balanced unused capacities. That is why the above modification ensures a solution's robustness.

In order to evaluate the performance of the proposed modification, it is necessary that some performance indexes be determined when demands are uncertain. Let U_D and U_R be the unmet demands percentages for the deterministic and the robust solutions respectively. In order to find them, first the demands are generated randomly in the interval $[d_i - \varepsilon d_i, d_i + \varepsilon d_i]$ and then, based on the vehicle capacity, the unmet demands are counted. For instance, in the problem explained in Table 3, the demands, considering 20% perturbation, would be 48, 45, 48, 29, 10, 40 and 54. In the optimum solution, the summations of the demands in the three routes would be 106, 81 and 69. So we would have 6 units as unmet demands related to Route 1 and U_D=6/257. In the robust solution, the summations of the demands in the routes are 96, 81 and 79. Therefore, we would have no unmet demands and, as a result, U_R =0/257. This procedure is repeated until the final values of U_D and U_R are found. Another index to be determined is Z_{DR} which indicates the extra cost of the robust solution compared to the deterministic one ($Z_{DR} = (Z_R - Z_D)/Z_D$). In this example Z_{DR} is equal to 0.02 ($= (235 - 230)/230$).

Next in the evaluation process, is to compare our heuristic results ($U_R(M)$, $Z_{DR}(M)$) and the exact robust optimum solutions ($U_R(S)$, $Z_{DR}(S)$) presented by Sungur et al. (2008). Although there are many results on SVRP, Sungur et al. (2008) are the only researchers who have investigated robust VRP. There are many differences between the robust VRP and the SVRP (see for example Ben-Tal and Nemirovski, 2000). Table 4 summarizes the results of the implementation of the proposed methods ($\varepsilon = 10\%$). In some cases, exact robust optimum solutions are not available (NA) in an acceptable amount of time because of the complexity of the related counterparts (Sungur et al., 2008). Since Sungur et al. (2008) have applied the exact algorithm to obtain the robust solution, their $U_R(S)$ is equal to zero for all instances; that is why we have not entered it in Table 4. In this table, data are computed based on a computer and environment similar to Sungur's et al. (2008), but the exact robust solution data have been extracted from Sungur et al. (2008).

Table 4. Comparison of the proposed heuristic method with exact robust solutions (Sungur et al., 2008)

No.	Sample	U_D	$Z_{DR}(S)$	$U_R(M)$	$Z_{DR}(M)$	No.	Sample	U_D	$Z_{DR}(S)$	$U_R(M)$	$Z_{DR}(M)$
1	An32k5	2.5	1.5	0	1	26	Bn52k7	2	0.7	0	0.3
2	An33k5	1.7	2.1	0	2	27	Bn56k7	3	3	0	3.2
3	An33k6	1.8	2.7	0	1.5	28	Bn57k7	5	IN	1.5	0.9
4	An34k5	0.3	1.5	0	0.6	29	Bn57k9	3.1	NA	0	1.2
5	An36k5	2.1	1.8	0	0.9	30	Bn63k10	3	NA	0.21	2.1

6	An37k5	1.6	2.7	0	1.9	31	Bn64k9	3.3	IN	0.9	3
7	An37k6	2.4	NA	0.2	0.5	32	Bn66k9	3.6	IN	0.91	2.1
8	An38k5	3.8	IN	0.67	1.5	33	Bn67k10	2.5	NA	0.87	3
9	An39k5	3.1	1.5	0.48	1.3	34	Bn68k9	3.9	NA	0.27	4.1
10	An44k6	2.2	1.5	0	0.9	35	Bn78k10	3.7	NA	0.7	2.4
11	An46k7	3.4	4.2	0.2	4.5	36	Pn23k8	3.4	IN	0.71	2.8
12	An60k9	2.1	NA	0.35	2.1	37	Pn40k5	1.2	0.7	0	0.3
13	Bn31k5	0.7	1.3	0	0.4	38	Pn45k5	3.3	2	0	1.1
14	Bn34k5	2.5	0.1	0	0.1	39	Pn50k7	2	2	0	1.5
15	Bn35k5	2.3	2.8	0	0.8	40	Pn50k8	3.9	IN	1.54	3
16	Bn38k6	2	0.1	0	0.1	41	Pn50k10	2.1	IN	0.82	2.1
17	Bn39k5	2.7	2.2	0	1.4	42	Pn51k10	3.2	IN	0.72	2
18	Bn41k6	2.4	4.2	0.4	2.1	43	Pn55k7	2	NA	0	0.4
19	Bn43k6	3.5	1.5	0	0.9	44	Pn55k8	1.9	NA	0	0.8
20	Bn44k7	3.6	4.6	0.3	3.5	45	Pn55k10	2.9	NA	0	1.1
21	Bn45k5	3.1	IN	0.79	1.2	46	Pn55k15	4.4	IN	1.4	2
22	Bn45k6	4.4	IN	0.84	1.1	47	Pn60k10	1.7	NA	0.68	1.8
23	Bn50k7	1.5	0.5	0	0.2	48	Pn60k15	2.3	NA	0.9	3
24	Bn50k8	3	2.7	0	1.1	49	Pn65k10	1.8	NA	0.94	2.5
25	Bn51k7	3.2	IN	0.83	0.9	50	Pn70k10	2.3	IN	1	2.7
							Average:	2.7	2^{A}	0.32	1.3^{A}

A: According to $Z_{DR}(S)$ column, since some data are infeasible or not available (IN or NA), the average is calculated only based on the existing data.

According to Table 4 data, applying paired T-test for the null hypothesis $H_0 : \mu_{Z_{DR}(M)} = \mu_{Z_{DR}(S)}$ against $H_1 : \mu_{Z_{DR}(M)} < \mu_{Z_{DR}(S)}$, based on the Minitab reported P-value (P-value=0), will result in the rejection of H_0. So, our method statistically leads to smaller costs than that of the exact method, proposed by Sungur et al. (2008). Also, in some samples, in Table 4, the exact method leads to infeasible solutions while ours has an appropriate solution.

Table 4 shows that our proposed method has two relative advantages in comparison with the robust exact solutions. First, in all cases it has yielded a solution while in Sungur et al. (2008) robust exact method, 26% of the cases have had no solutions, 26% have had infeasible counterparts and 48% of the cases have yielded a robust solution. Second, the average for the values of $Z_{DR}(M)$ equals 1.3 while this value for $Z_{DR}(S)$ equals 2 which shows that in our proposed PSO method, the extra cost is 7% less than that of the Sungur's et al. (2008). However, the exact robust method, for cases with available solutions, meets all uncertain demands while our method misses some demands (in Table 4, the average of unmet demands is 0.32%). In such cases, these two methods should be compared economically.

As was explained before, El Ghaoui (2003) illustrates the robust solution by a conceptual example which is depicted in Figure 1. In our method, we have proposed a new heuristic counterpart area (the red bound) which is larger than the exact counterpart shown in his figure. Since the red area is larger than the exact counterpart, sometimes it includes areas affected by perturbation. That is why on the one hand we have some unmet demands, but on the other hand, in some cases $Z_{DR}(M)$ is smaller than $Z_{DR}(S)$. Therefore, although the exact robust algorithm gives solutions without any unmet demands, the proposed method (even with some unmet demands) has much better solutions. Yet, in practical cases, we should compare the cost of unmet demands in the proposed method with the extra cost of the exact robust method to see which solution is more economical.

5. Conclusions

In this paper, a novel Particle Swarm Optimization (PSO) approach is applied to solve the CVRP having stochastic demands with no known distributions. Since PSO solves problems with continuous variables and the VRP is a combinatorial optimization problem, a decoding method is needed to apply PSO to solve the VRP. We have developed a novel decoding method for interpreting PSO solutions for the VRP. The proposed M1 decoding method includes three local search operators in a VNS loop in order to significantly improve the quality of the solutions. It has been tested against two recent and state-of-the-art decoding methods from Ai and Kachitvichyanukul, (2009b). The results clearly show that our proposed approach is a superior and preferable PSO method in some CVRP instances. The whole proposed PSO approach has also been tested for the Stochastic VRP (SVRP). Results show that the solutions of this algorithm are applicable in larger scale problems and resist perturbations to an acceptable extent. Although the exact robust algorithm gives solutions without any unmet demands, our proposed method (even with some unmet demands) has less robustness cost. Also, in all cases, the proposed method has produced a feasible solution while the other method has not (in many cases).

Our intention for future research is to actually test the proposed PSO in a production environment with a real SVRP and to actually measure unmet vs. robustness costs. A close study of the performance of the proposed PSO parameters can be the subject of further researches.

References

[1] Ai, J., & Kachitvichyanukul, V. (2009a). A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research, 36* (5), 1693–1702.

[2] Ai, J., & Kachitvichyanukul, V. (2009b). Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. *Computers & Industrial Engineering, 56* (1), 380–387.

[3] Ai, J., & Kachitvichyanukul, V. (2009c). A Particle Swarm Optimisation for Vehicle Routing Problem with Time Windows. *International Journal of Operational Research, 6* (4), 519–537.

[4] Ai, J., & Kachitvichyanukul, V. (2008). A Study on Adaptive Particle Swarm Optimization for Solving Vehicle Routing Problems. *The 9th Asia Pacific Industrial Engineering and Management Systems Conference*, December, Bali, Indonesia.

[5] Angeline, P. J. (1998). Evolutionary optimization versus particle swarm optimization: philosophy and performance difference. *7th Annual Conference on Evolutionary Programming*, March, San Diego, CA, USA.

[6] Augerat, P., Belenguer, J., Benavent, E., Corbern, A., Naddef, D., & Rinaldi, G. (1995). *Computational results with a branch and cut code for capacitated vehicle routing problem.* Research Report RR 949-M, University Joseph Fourier, France.

[7] Ben-Tal, A., & Nemirovski, A. (1998). Robust convex optimization. *Mathematics of Operations Research, 23* (4), 769–805.

[8] Ben-Tal, A., & Nemirovski, A. (1999). Robust solutions to uncertain linear programs. *Operation Research Letters, 25* (1), 1–13.

[9] Ben-Tal, A., & Nemirovski, A. (2000). Robust solutions of linear programming problems contaminated with uncertain data. *Mathematic Programming*, 88, 411-424.

[10] Bertsimas, D. J. (1992). A vehicle routing problem with stochastic demand. *Operations Research , 40* (3), 574–585.

[11] Chen, A., Yang, G., & Wu, Z. (2006). Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. *Journal of Zhejiang University Science, 7* (4), 607–614.

[12] Clark, A. J., & Scarf, H. (1960). Optimal policies for a multi-echelon inventory problem. *Management Science, 6* (4), 475–490.

[13] Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1), 58-73.

[14] Cordeau, J.-F., Gendreau, M., Hertz, A., Laporte, G., & Sormany, J.-S. (2005). New Heuristics for the Vehicle Routing Problem. In A. Langevin, & D. Riopel (Eds.), *Logistics Systems: Design and Optimization*, 279–297. Springer. New York.

[15] Dessouky, M., Ordonez, F., Jia, H., & Shen, Z. (2006). Patient flow: reducing delay management in health care delivery. In R. Hall (Ed.), *Rapid Distribution of Medical Supplies*. Springer. New York.

[16] Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science, 6* (1), 80–91.

[17] Dror, M. (1993). Modeling vehicle routing with uncertain demands as a stochastic program. *European Journal of Operational Research , 64* (3), 432–441.

[18] El Ghaoui, L., Oustry, F., & Lebret, H. (1998). Robust solutions to uncertain semi-definite programs. *SIAM Journal on Optimization, 9* (1), 33–52.

[19] El Ghaoui, L. (2003). Retrieved from Robust optimization and applications:www.ima.umn.edu/talks/workshops/3-11.2003/elghaoui

[20] Erbao, C., & Mingyong, L. (2009). A hybrid differential algorithm to vehicle routing problem with fuzzy demand. *Journal of Computational and Applied Mathematics, 231* (1), 302–310.

[21] Gendreau, M., Laporte, G., & Seguin, R. (1995). An Exact Algorithm for The Vehicle Routing Problem with Stochastic Demands and Customers. *Transportation Science, 29* (2), 143–155.

[22] Gendreau, M., Laporte, G., & Seguin, R. (1996a). A Tabu Search Heuristic for the Vehicle Routing Problem with Stochastic Demands and Customers. *Operations Research, 44* (3), 469–477.

[23] Gendreau, M., Laporte, G., & Seguin, R. (1996b). Stochastic vehicle routing. *European Journal of Operational Research, 88* (1), 3–12.

[24] Golden, B. L., & Yee, J. R. (1979). A framework for probabilistic vehicle routing. *AIIE Transactions*, *11* (2), 109–112.
[25] Graves, S. C., Rinooy Kan, A.H.G., & Zipkin, P.H. (1993). *Logistics of Production and Inventory: Handbooks in OR & MS* (Vol. 4). Elsevier. Amsterdam.
[26] Hansen, P., & Mladenovic, N. (2003). Variable neighborhood search. In F. Glover, & G. Kochenberge (Eds.), *Handbook of Metaheuristics* (pp. 145–184). Kluwer. Boston.
[27] Jaillet, P., & Odoni, A. (1988). The Probabilistic Vehicle Routing Problem, In B. L. Golden, & A. A. Assad, (Eds.), *Vehicle Routing; Methods and Studies* (pp. 293-318). North Holland. Amsterdam.
[28] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *IEEE International Connference Neural Networks*, November, Perth, Australia, 1942–1948.
[29] Kindervater, G. A. P., & Savelsbergh, M. W. P. (1997). Vehicle routing: handling edge exchanges. In E. H. Aarts, & J. K. Lenstra (Eds.), *Local Search in Combinatorial Optimization* (pp. 311-336). John Wiley & Sons. UK.
[30] Laporte, G. (1992). The Vehicle Routing Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, *59* (3), 345–358.
[31] Laporte, G., Gendreau, M., Potvin, J. Y., & Semet, F. (2000). Classical amd Modern Heuristics for the vehicle routing problem. *International Transactions in Operational Research*, *7*, 285–300.
[32] Lenstra, J. K., & Rinnooy Kan, A.H.G. (1981). Complexity of vehicle routing and scheduling problems. *Networks, 11* (2), 221–227.
[33] Liu, B., & Lai, K. K. (2002). Stochastic programming models for vehicle routing problems. *Asian Information Science Life, 1* (1), 13–28.
[34] Markovic, H., Cavar, I., & Caric, T. (2005). Using data mining to forecast uncertain demands in Stochastic Vehicle Routing Problem. *13th International Symposium on Electronics in Transport (ISEP),* May, Slovenia.
[35] Novoa, C., Storer, R. (2009). An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research, 196* (2), 509–515.
[36] Önüt, S., Tuzkaya, R. U., & Doğaç, B. (2008). A particle swarm optimization algorithm for the multiple level warehouse layout design problem. *Computers & Industrial Engineering, 54* (4), 783–799.
[37] Ozcan, E., & Mohan, C. K. (1999). Particle swarm optimization: surfing the waves. *Proceedings of the IEEE Congress on Evolutionary Computation*, July, Washington D.C., USA.
[38] Peram, T., Veeramachaneni, K., & Mohan, C. K. (2003). Fitness-distance-ratio based particle swarm optimization. *Proceedings of the IEEE Swarm Intelligence Symposium*, Indianapolis, Indiana, USA.
[39] Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications, 1*, 1–10.
[40] Pirkwieser, S., & Raidl, G. R. (2008). A variable neighborhood search for the periodic vehicle routing problem with time windows. *Proceedings of the 9th EU/Meeting on Metaheuristics* for *Logistics and Vehicle Routing*, October, Troyes, France.
[41] Polacek, M., Hartl, R., Doerner, K., & Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of Heuristics, 10* (6), 613–627.
[42] Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, *177* (3), 2033–2049.
[43] Ruiz, R., & Stützle, T. (2008). An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research , 87* (3), 1143–1159.

[44] Shen, Z., Ordóñez, F., & Dessouky, M. M. (2009). The Stochastic Vehicle Routing Problem for Minimum Unmet Demand. In W. Chaovalitwongse, K. Furman & P. Pardalos (Eds.), *Optimization and Logistics Challenges in the Enterprise* (pp. 349–371). Springer. New York.

[45] Shi, X. H., Liang, Y. C., Lee, H. P., Lu, C., & Wang, Q. X. (2007). Particle swarm optimization based algorithms for TSP and generalized TSP. *Information Processing Letters, 103* (5), 169–176.

[46] Stewart, W. R., & Golden, B. L. (1983). Stochastic vehicle routing: A comprehensive approach. *European Journal of Operational Research, 14* (4), 371–385.

[47] Sungur, F., Ordóñez, F., & Dessouky, M. M. (2008). A Robust Optimization Approach for the Capacitated Vehicle Routing Problem with Demand Uncertainty. *IIE Transactions, 40* (5), 509–523.

[48] Tao, Z., Chunmei, Z., Yuejie, Z., & Chuoya, Y. (2008). A Mixed PSO algorithm for the VRPSPD. Control *and Decision Conference*, May, Chinese, (pp. 4017–4021).

[49] Tillman, F. (1969). The multiple terminal delivery problem with probabilistic demands. *Transportation Science, 3* (3), 192–204.

[50] Veeramachaneni, K., Peram, T., Mohan, C., & Osadciw, L. (2003). Optimization using particle swarm with near neighbor interactions. *Genetic and Evolutionary Computation Conferenc,* July, Chicago, USA.

[51] Yang, S. Y., Wang, M., & Jiao, L. C. (2004). A Quantum Particle Swarm Optimization. *Congress on Evolutionary Computation*, June, Portland, USA, 320–324.