Applied Soft Computing 13 (2013) 1693–1704

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

# Particle Swarm Optimization for the Vehicle Routing Problem with Stochastic Demands

Yannis Marinakis*, Georgia-Roumbini Iordanidou, Magdalene Marinaki

*Technical University of Crete, Department of Production Engineering and Management, University Campus, 73100 Chania, Crete, Greece*

ARTICLE INFO

*Article history:*
Received 6 October 2011
Received in revised form 2 December 2012
Accepted 11 January 2013
Available online 22 January 2013

*Keywords:*
Particle Swarm Optimization
Path relinking
Vehicle Routing Problem with Stochastic Demands

ABSTRACT

This paper introduces a new hybrid algorithmic approach based on Particle Swarm Optimization (PSO) for successfully solving one of the most popular supply chain management problems, the Vehicle Routing Problem with Stochastic Demands (VRPSD). The VRPSD 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. A number of different variants of the PSO are tested and the one that performs better is used for solving benchmark instances from the literature.



## 1. Introduction

The **Vehicle Routing Problem (VRP)** or the **Capacitated Vehicle Routing problem (CVRP)** is often described as the problem in which vehicles based on a central depot are required to visit geographically dispersed customers in order to fulfill known customer demands. The vehicle routing problem was first introduced by Dantzig and Ramser [13]. Since then a number of variants of the classic Vehicle Routing Problem has been proposed in order to incorporate more constraints like time windows, multi-depot, stochastic or dynamic demands. The reader can find more detailed descriptions of the algorithms proposed for the CVRP and its variants in the survey papers [6,7,16,18,19,35,34,40,55] and in the books [21,22,46,59].

In Stochastic Vehicle Routing Problems (SVRPs) one parameter of the problem, like customers, customers' demands or customers' travel and service times, is a stochastic variable that follows a known (or unknown) probability distribution. Stochastic VRPs differ from the Capacitated Vehicle Routing Problems in several fundamental aspects. The concept of a solution is different, several fundamental properties of deterministic VRPs no longer hold in the stochastic case, and solution methodologies are considerably more intricate. Since they combine the characteristics of stochastic and integer programs, SVRPs are often regarded as computationally intractable [17].

The Vehicle Routing Problem with Stochastic Demands (VRPSD) 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 demand is known only when the vehicle arrives to them (this is the main difference of the VRPSD from the Capacitated VRP where all the customer demands are known beforehand). A route begins from the depot and visits each customer exactly once and returns to the depot. This is called an a priori tour. The a priori tour can be seen as a template for the visiting sequence of all customers. In a given instance, the customers should be visited based on the sequence of the a priori tour but the final route includes, also, returns to the depot when the vehicle needs replenishment. The nodes from which the vehicle returns to the depot are stochastic points [53].

Some of the applications of a SVRP are [17,50,53]:

- delivery of home heating oil,
- delivery of petrol to petrol stations,
- garbage collection,
- sludge disposal,
- beer and soft drinks distribution,
- the provision of bank automates with cash,
- the collection of cash from bank branches,
- collect milk from different producers,
- distribution of products in grocery stores.

Due to the difficulty of the Vehicle Routing Problem with Stochastic Demand and to the fact that it is an NP-hard problem the instances with a large number of customers cannot be solved in optimality within reasonable time. For this reason, a number of

* Corresponding author. Tel.: +30 28210 37288; fax: +30 28210 69410.
*E-mail addresses:* marinakis@ergasya.tuc.gr (Y. Marinakis), rubinior@yahoo.gr (G.-R. Iordanidou), magda@dssl.tuc.gr (M. Marinaki).

1568-4946/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2013.01.007

approximation techniques (metaheuristic and evolutionary algorithms) were proposed for the solution of the problem. In this paper, the Vehicle Routing Problem with Stochastic Demand is solved using one of these algorithms, the Particle Swarm Optimization algorithm. **Particle Swarm Optimization (PSO)** is a population-based swarm intelligence algorithm that was originally proposed by Kennedy and Eberhart [30]. PSO simulates the social behavior of social organisms by using the physical movements of the individuals in the swarm. Its mechanism enhances and adapts to the global and local exploration. Most applications of PSO have concentrated on the optimization in continuous space while some work has been done to the discrete optimization [31,52]. Recent complete surveys for the Particle Swarm Optimization can be found in [1,2,11,47]. The Particle Swarm Optimization (PSO) is a very popular optimization method and its wide use, mainly during the last years, is due to the number of advantages that this method has, compared to other optimization methods. Some of the key advantages are that this method does not need the calculation of derivatives that the knowledge of good solutions is retained by all particles and that particles in the swarm share information between them. PSO is less sensitive to the nature of the objective function, can be used for stochastic objective functions and can easily escape from local minima. Concerning its implementation, PSO can easily be programmed, has few parameters to regulate and the assessment of the optimum is independent of the initial solution.

Thus, in this paper, we demonstrate how a nature inspired intelligent technique, the Particle Swarm Optimization (PSO) [30], two simple local search metaheuristics (2-opt and 3-opt) and the Path Relinking strategy [20] can be incorporated in a hybrid scheme, in order to give very good results for the Vehicle Routing Problem with Stochastic Demands (VRPSD). A number of different variants of Particle Swarm Optimization are tested and the one that gives the best performance in a number of tests is used for the whole set of the test instances and for the comparisons with other Evolutionary Optimization techniques. Particle Swarm Optimization algorithms, due to their simplicity in coding and their global and local exploration abilities, are usually applied with remarkable results in continuous optimization problems. In this paper, we focus in the way a PSO algorithm can easily and efficiently be applied in a classic Combinatorial Optimization Problem, like the Vehicle Routing Problem with Stochastic Demands. The main advantage of the application of a PSO algorithm in the VRPSD is that, in contrary to others metaheuristics, there are only two variables for each member of the population that will have to be calculated in each iteration, the position and the velocity. As we would like to increase more the efficiency of the Particle Swarm Optimization we incorporate the local search metaheuristics and the path relinking strategy.

The rest of the paper is organized as follows: In Section 2, the formulation of the Vehicle Routing Problem with Stochastic Demands is given while in Section 3 a literature review is presented. In Section 4, an analytical description of the proposed algorithm is given. In Section 5, the computational results of the algorithm are presented and analyzed. Finally, in the last section some general conclusions and the future research are given.

## 2. The Vehicle Routing Problem with Stochastic Demands

A solution of the Vehicle Routing Problem with stochastic demands is a permutation of the customers starting from the depot. This is called an a priori tour [4]. To find the route that makes the vehicle, an initial path which starts from the depot is considering. Depending on the demand of the next customer in the a priori tour it is deciding if the vehicle will return to the depot for restocking or if it will continue to the next customer. Many times although the expected demand of the customer is less than the vehicle's load, it is chosen the return of the vehicle to the depot for replenishment, and this is called 'preventive restocking'. The 'preventive restocking' aims to avoid the risk of the vehicle to go to the next customer without having enough load to satisfy him. If this is happened the vehicle will have to go back to the depot and then return to the same customer.

The Vehicle Routing Problem with Stochastic Demands is defined on a complete graph $G = (V, A, D)$, where:

- $V = \{0, 1, \ldots, n\}$ is the set of nodes where node *0* is the depot,
- $A = \{(i, j) : i, j \in V, i \neq j\}$ is the set of arcs,
- $D = \{d_{ij} : i, j \in V, i \neq j\}$ is the travel distance between node $i$ and $j$.

A vehicle with capacity $Q$ has to deliver products to the customers based on their demand minimizing the total expected length based on the following [4]:

- The customers' demands ($\xi_i$, $i = 1, \ldots, n$) are stochastic variables independently distributed with known distributions.
- The real demand of each customer is known only when the vehicle arrives to the customer.
- The demand $\xi_i$, $i = 1, \ldots, n$ does not exceed the vehicle's capacity $Q$, and follows a discrete probability distribution $p_{ik} = Prob(\xi_i = k)$, $k = 0, 1, 2, \ldots, K \leq Q$.

The objective function of the problem which is the expected distance (length) of the a priori tour that a vehicle will travel is calculated by the following equation [4]:

$$f_j(q) = Minimum\{f_j^p(q), f_j^r(q)\} \tag{1}$$

where

$$f_j^p(q) = d_{j,j+1} + \sum_{k \leq q} f_{j+1}(q-k)p_{j+1,k} + \sum_{k>q} [2d_{j+1,0} + f_{j+1}(q+Q-k)]p_{j+1,k} \tag{2}$$

$$f_j^r(q) = d_{j,0} + d_{0,j+1} + \sum_{k=1}^{K} f_{j+1}(Q-k)p_{j+1,k} \tag{3}$$

with boundary condition

$$f_n(q) = d_{n,0}, \; q \in L_n \tag{4}$$

where

- $s = (0, 1, \ldots, n)$ is an a priori tour,
- $q$ is the remaining load of the vehicle after the completion of the service in customer $j$,
- $f_j(q)$ is the expected cost from the customer $j$ onward (thus, if $j = 0$ then $f_0(q)$ denotes the expected cost of the a priori tour),
- $f_j^p(q)$ is the expected cost of the route when the vehicle does not return to the depot but goes to the next customer, and,
- $f_j^r(q)$ is the expected cost when the vehicle returns to the depot for preventive restocking.

The random behavior of customer demands could cause an expected feasible solution to become infeasible if the final demand of any route exceeds the actual vehicle capacity. This situation is referred to as route failure, and when it occurs, some corrective actions must be introduced to obtain a new feasible solution. Thus, in order to avoid a route failure, the optimum choice is the choice of a threshold value [61]. If the residual load after customer $j$ is

greater or equal to $h_j$, then it is better to move to the next customer, otherwise it is better to return to the depot for preventive restocking.

## 3. Literature review

Many studies have been done to solve the vehicle routing problem with stochastic parameters. In the following, some of them will be mentioned. It is worth noting that the first who proposed an algorithm for the Stochastic Vehicle Routing Problem was Tillman in 1969 [58]. An early survey paper with presentation of the most important models by that time is presented in [53]. In [14], a class of stochastic vehicle routing problems with random demands is presented in which the number of potential failures per route is restricted either by the data or the problem constraints. A literature review with the models and applications till 1996 is presented in [17]. Protonotarios et al. [48] dealt with SVRP, where they wanted to minimize transportation costs and maximize customers' satisfaction in large-scale problems, having as constraints the capacity of the vehicles, time windows for customers' service and working hours per day for drivers. The customers' demand is stochastic. They proposed a genetic algorithm to solve the problem. Yang et al. [61] use the customers' demand as a stochastic parameter. The difference here is that the restocking points may arise before a stockout actually occurs. They solved the problem with two heuristic algorithms to construct both single and multiple routes that minimize total travel cost. In [50], two neuro-dynamic programming algorithms (optimistic approximate policy iteration and a rollout policy) are compared for the solution of the vehicle routing problem where customers' demands are uncertain. Kenyon and Morton [33] considered stochastic vehicle routing problems on a network with random travel and service times. One or more vehicles are available to be routed through the network to service each node. Two versions of the model are developed based on alternative objective functions. They provided bounds on optimal objective function values and conditions under which reductions to simpler models can be made. Their solution method embeds a branch-and-cut scheme within a Monte Carlo sampling-based procedure.

Guo and Mac [24] solved a SVRP with a genetic algorithm, having as stochastic parameters the customers' demand and the presence of customers at the delivery point. In [3], a dynamic Vehicle Routing Problem with Time Windows (VRPTW) with stochastic customers is presented, where the goal is to maximize the number of serviced customers. It presents a multiple scenario approach (MSA) that continuously generates routing plans for scenarios including known and future requests. Decisions during execution use a distinguished plan chosen, at each decision, by a consensus function. Hvattum et al. [26] suggest an algorithm in which the basic idea is to solve different scenarios and, finally, the use of such features of the scenarios to build a good plan. The stochastic parameters of their model were the customers and the demands. Their goal was to minimize the vehicles and the cost of the routes. Reimann [49] used the ant colony optimization algorithm to solve the problem with the demand of customers as a stochastic parameter. Shen et al. [51] dealt with routing vehicles to service large-scale emergencies such as natural disasters and terrorist attacks. They decomposed the problem into two stages: a planning stage and an operational stage. In the planning stage, they aimed to generate the routes well in advance of any emergencies and in the operational stage, they decided the delivery quantity with the planned routes and made adjustments to the routes if necessary at the time of the emergency when more information was revealed.

Bianchi et al. [4] used a number of hybrid metaheuristic algorithms for solving the SVRP (simulated annealing, tabu search, repeated local search, ant colony optimization and evolutionary algorithms). Yan et al. [60] applied a simulation technique with strategies based on connections and paths to develop two heuristic algorithms to solve the SVRP. They tested their model on a city bus in Taiwan. A branch and price algorithm for the capacitated Vehicle Routing Problem with Stochastic Demands is presented in [9]. In [56], initially, a multiobjective formulation of the problem is given and, then, a multiobjective evolutionary algorithm that incorporates two VRPSD-specific heuristics for local exploitation and a route simulation method to evaluate the fitness of solutions, is proposed. Haugland et al. [25] solved the SVRP with stochastic demands using a Tabu Search algorithm and a multi-opening heuristic algorithm. Li et al. [37] solved the SVRP with time windows and with stochastic parameters (the time the vehicle needs to complete a route, the distributions and the service time of customers) using a tabu search algorithm. In [5], a survey of the appropriate metaheuristics to solve a wide class of combinatorial optimization problems under uncertainty is given. In [45], approximate dynamic programming algorithms are presented for the single Vehicle Routing Problem with Stochastic Demands from a dynamic or reoptimization perspective. Mendoza et al. [42] dealt with the multi-compartment vehicle routing problem that consists of designing transportation routes to satisfy the demands of a set of customers for several products that must be loaded in independent vehicle compartments due to incompatibility constraints. The problem was solved using a memetic algorithm.

In [43], the routing of a single vehicle that delivers products and picks up items with stochastic demand is presented. A suitable dynamic programming algorithm is proposed to determine the minimum expected routing cost. In [27], the authors used a transformation of the Stochastic Vehicle Routing Problem in a small set of Capacitated Vehicle Routing Problems. In this paper, the CVRP instances are obtained from the original VRPSD instance by assigning different values to the level of safety stocks that routed vehicles must employ to deal with unexpected demands. The methodology also makes use of Monte Carlo simulation (MCS) to obtain estimates of the reliability of each a priori solution. In [36], the capacitated Vehicle Routing Problem with Stochastic Demands and time windows, that is an extension of the capacitated Vehicle Routing Problem with Stochastic Demands, is presented and formulated. Also, an adaptive large neighborhood search heuristic for its solution is proposed.

In [23], neighborhood structures for heuristic search applicable to a general class of vehicle routing problems (VRPs) are presented. The authors proposed a methodology that utilize a cyclic-order solution encoding, which maps a permutation of the customer set to a collection of many possible VRP solutions. Utilizing a simulated annealing framework, they demonstrated the potential of cyclic-order neighborhoods to facilitate the discovery of high quality a priori solutions for the vehicle routing problem with stochastic demand (VRPSD). In [28] the authors discussed how Parallel and Distributed Computing Systems can be employed to efficiently solve the VRPSD. A number of scenarios with different levels of safety stocks have been applied and, then, the algorithm solves each scenario by integrating Monte Carlo simulation inside a heuristic-randomization process.

## 4. Particle Swarm Optimization for the Vehicle Routing Problem with Stochastic Demands

### 4.1. General description

The Particle Swarm Optimization (PSO) algorithm was proposed by Kennedy and Eberhart [30–32] to simulate the social behavior of social organisms such as bird flocking and fish schooling. This method has been identified as very useful in many problems. The

reason is that the implementation is easy and it gives good results, especially in problems with continuous variables.

Some of the advantages of this method is that:

- it has memory which is important because the information from past good solutions passes on to future generations,
- there is cooperation between particles (solutions) of the swarm because they work together to create solutions.

In this section, the proposed Particle Swarm Optimization (PSO) algorithm for the solution of the Vehicle Routing Problem with Stochastic Demands is given. 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. One of the key issues in designing a successful PSO for the Vehicle Routing Problem with Stochastic Demands is to find a suitable mapping between Vehicle Routing Problem with Stochastic Demands solutions and particles in PSO. Each particle is recorded via the path representation of the tour, that is, via the specific sequence of the nodes (see Section 4.2). The position of each individual (called particle) is represented by a $d$-dimensional vector in problem space $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$, $i = 1, 2, \ldots, N$ ($N$ is the population size and $d$ is the number of the vector's dimension), and its performance is evaluated on the predefined fitness function ($f(x_{ij})$) (see Section 4.3). 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 ($gbest_j$).

In this paper, a number of different variants of Particle Swarm Optimization are used in order to find the one that works better for the Vehicle Routing Problem with Stochastic Demands. The differences of these variants are mainly concern in the calculation of the velocities. In Section 4.4, all these variants are presented and analyzed.

Initially, the routes of each particle are created with random node sequence and the particles' velocities are initialized with zeros. In every variant, the position of a particle changes using the following equation:

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{5}$$

where $t$ is the iterations' counter. A particle's best position ($pbest_{ij}$) in a swarm is calculated from the equation:

$$pbest_{ij} = \begin{cases} x_{ij}(t+1), & \text{if } f(x_{ij}(t+1)) < f(x_{ij}(t)) \\ pbest_{ij}, & \text{otherwise} \end{cases} \tag{6}$$

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

$$gbest_j \in \{pbest_{1j}, pbest_{2j}, \ldots, pbest_{Nj} | f(gbest_j)\} = min\{f(pbest_{1j}), f(pbest_{2j}), \ldots, f(pbest_{Nj})\} \tag{7}$$

A local search strategy is used in order to improve the solutions produced from the Particle Swarm Optimization algorithm (see Section 4.5). In each iteration of the algorithm the optimal solution of the whole swarm and the optimal solution of each particle are kept. The algorithm stops when a maximum number of iterations has been reached. A pseudocode of the proposed algorithm is presented in Table 1.

**Table 1**
Particle Swarm Optimization for the Vehicle Routing Problem with Stochastic Demands.

*Initialization*
**Select** the variant of the PSO algorithm
**Select** the number of swarms
**Select** the number of particles in each swarm
**Choose** the demands' deviation
**Initialize** the routes with a random way
**Convert** particles' positions in continuous form
**Initialize** the position and velocity of each particle
**Calculate** the initial fitness function of each particle
**Find** the best solution of each particle
**Find** the best particle of the entire swarm
*Main Phase*
**Do while** the maximum number of iterations has not be reached
  **Calculate** the velocity of each particle
  **Calculate** the new position of each particle
  **Convert** particles' positions in integer form
  **Calculate** the new fitness function of each particle
  **Improve** the solutions with 2-opt, 3-opt, Path Relinking
  **Update** the best solution of each particle
  **Find** the best particle of the whole swarm
  **Convert** particles' positions in continuous form
**Enddo**
**Return** the best particle (best solution)

### 4.2. Solution representation

All the solutions (particles) are represented with the path representation of the tour. For example, if we have a particle with eight nodes a possible path representation is the following:

1 8 5 3 7 2 4 6

In case these routes do not start with node 1, we find and put node 1 at the beginning of the route as it is necessary for the calculation of the fitness function (see Section 4.3). As the calculation of the velocity of each particle is performed by Eqs. (8)–(21) (see below), the above mentioned representation will have to be transformed appropriately. We transform each element of the solution into a floating point interval (0,1], calculate the velocities and the positions of all particles and, then, convert back the particles' positions into the integer domain using relative position indexing [38]. Thus, initially, we divide each element of the solution by the vector's largest element. For the previous example, the particle becomes:

0.125 1 0.625 0.375 0.875 0.25 0.5 0.75

After the calculation of the positions, the elements of the positions' vector are transformed back into the integer domain by assigning the smallest floating value to the smallest integer (0.18 to 1 in the following example), the next floating value to the next integer (0.28 to 2 in the following example) and so on until the largest floating value to the largest integer (0.93 to 8 in the following example). Thus, if the positions' vector of the previous particle has become:

0.56 0.93 0.28 0.789 0.18 0.64 0.316 0.533

the backward transformation gives

5 8 2 7 1 6 3 4

For the calculation of the fitness function, the node 1 (the depot) is needed to be in the first position of the vector, thus, the previous vector becomes:

1 6 3 4 5 8 2 7

### 4.3. Calculation of the fitness function

Concerning the fitness function, it should be noted that in Vehicle Routing Problem with Stochastic Demands, the fitness of each particle is related to the expected length of the a priori tour and, since the problem that we deal with is a minimization problem, if a feasible solution has a high objective function value, then, it is characterized as an unpromising solution candidate.

The assessment of the route cost begins from the route end. In the beginning, the probability of the demand of each customer to take a particular value is stored in a variable. This probability depends on the value of the demand's deviation. For example, if the demand's deviation is $r$ and the real demand is $R$ (where $r \leq R$), the probability of the demand is $1/(2r+1)$ because the demand can take $2r+1$ values (i.e., $R-r, R-(r-1), \ldots, R, \ldots, R+(r-1), R+r$) and the probability of the demand to take one of these values is the same. The cost from the last node to the depot can be assessed directly as it does not depend on the customer demand. The fitness function is calculated using Eqs. (1)–(4). Based on the fitness function the vehicle either returns to the depot for replenishment or it proceeds to the next customer.

### 4.4. Velocity equations

In the literature, a number of different variants of the Particle Swarm Optimization have been proposed for the calculation of the velocities. In this paper, some of them are used in order to find which is the most suitable for the Vehicle Routing Problem with Stochastic Demands. In the following, an analytical description of the variants used in this paper is presented (these variants are denoted by PSO1, PSO2, . . ., PSO8).

- **PSO1: Classic Particle Swarm Optimization** [30]:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 rand_1(pbest_{ij} - x_{ij}(t)) + c_2 rand_2(gbest_j - x_{ij}(t)) \quad (8)$$

where $c_1$ and $c_2$ are the acceleration coefficients, $rand_1$ and $rand_2$ are two random variables in the interval [0, 1]. 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 [30]. If $c_1 = c_2 = 0$ the particles are directed where their velocity indicates, if $c_1 > 0$ and $c_2 = 0$, then, each particle is influenced only by its previous moves and not from the other particles in the swarm and if $c_2 > 0$ and $c_1 = 0$, all the particles follow the best particle. Most of the time the researchers select $c_1 = c_2$. In this case, the particle is influenced equally by both factors. Also, sometimes the values of $c_1$ and $c_2$ are changed so that the influence of the two factors can vary during the iterations. In our algorithm, we use the following definition for $c_1$ and $c_2$. Let $c_{1,min}$, $c_{1,max}$, $c_{2,min}$, $c_{2,max}$ be the minimum and maximum values that $c_1, c_2$ can take, respectively, then:

$$c_1 = c_{1,min} + \frac{c_{1,max} - c_{1,min}}{iter_{max}} \times t \quad (9)$$

$$c_2 = c_{2,min} + \frac{c_{2,max} - c_{2,min}}{iter_{max}} \times t \quad (10)$$

where $t$ is the number of current iteration and $iter_{max}$ the maximum number of the iterations. In the first iterations of the algorithm, the values of $c_1$ and $c_2$ are small and, then, they increase until they reach to their maximum values. The advantage of this definition for $c_1$ and $c_2$ is that in the first iterations there is a great freedom of movement in the particles' (solutions') space in order to find the minimum quickly.

- **PSO2. Inertia Particle Swarm Optimization** [52]:

$$v_{ij}(t+1) = w v_{ij}(t) + c_1 rand_1(pbest_{ij} - x_{ij}(t)) + c_2 rand_2(gbest_j - x_{ij}(t)) \quad (11)$$

The difference of this equation from Eq. (8) is the use of the inertia weight $w$. The inertia weight is used to control the impact of previous histories of velocities on the current velocity. The particle adjusts its trajectory based on information about its previous best performance and the best performance of its neighbors. The inertia weight $w$ is, also, used to control the convergence behavior of the PSO. A number of different alternatives for the definition of $w$ have been proposed. These alternatives vary from constant values to different ways of increasing or decreasing of $w$ during the iterations. In this paper, in order to exploit more areas in the solution space, the inertia weight $w$ is updated according to the following equation:

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times t \quad (12)$$

where $w_{max}, w_{min}$ are the maximum and minimum values of inertia weight.

- **PSO3. Constriction Particle Swarm Optimization 1** [12]:

$$v_{ij}(t+1) = \chi(v_{ij}(t) + c_1 rand_1(pbest_{ij} - x_{ij}(t)) + c_2 rand_2(gbest_j - x_{ij}(t))) \quad (13)$$

where

$$\chi = \frac{2}{|2 - c - \sqrt{c^2 - 4c}|} \quad \text{and} \quad c = c_1 + c_2, c > 4 \quad (14)$$

The difference of this variant from the two previous variants is the use of a limiting factor ($\chi$), called constriction factor. The constriction factor is used in order to prevent explosion, to ensure convergence of the algorithm and to eliminate the factors that limit the velocities of the particles.

- **PSO4. Constriction Particle Swarm Optimization 2** [15]:

In this variant, the velocity equation is the same as in the Constriction Particle Swarm Optimization (Eq. (13)) but the constriction factor is calculated by [15]:

$$\chi = \frac{2k}{|2 - c - \sqrt{c^2 - 4c}|} \quad \text{and} \quad c = c_1 + c_2, c > 4 \quad (15)$$

where $k \in (0, 1]$. When $k$ takes values close to zero the algorithm's exploitation abilities are increased while when it takes values close to one the algorithm's exploration abilities are increased. In the last few years extensive using of all variants of PSO has proved that the use of constriction factor gives much better results than the other variants of the algorithm.

- **PSO5. Constriction Particle Swarm Optimization 3** [15]:

A simpler variant of the velocity equation (Eq. 13) is [15]:

$$v_{ij}(t+1) = \chi(v_{ij}(t) + c(p_{mj} - x_{ij}(t))) \quad (16)$$

where

$$\chi = \frac{2}{|2 - c - \sqrt{c^2 - 4c}|} \quad \text{and} \quad c = c_1 + c_2, c > 4 \quad (17)$$

and

$$p_{mj} = \frac{c_1 pbest_{ij} + c_2 gbest_j}{c} \quad (18)$$

- **PSO6. Cognition-only Particle Swarm Optimization** [29]:

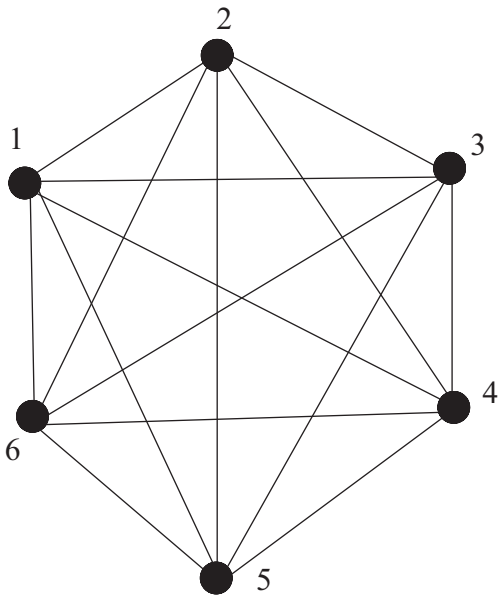


**Fig. 1.** Star topology.

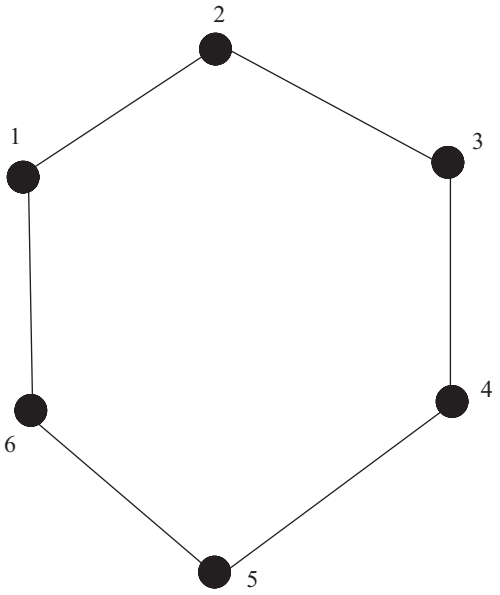


**Fig. 2.** Ring topology with 3 neighbors.

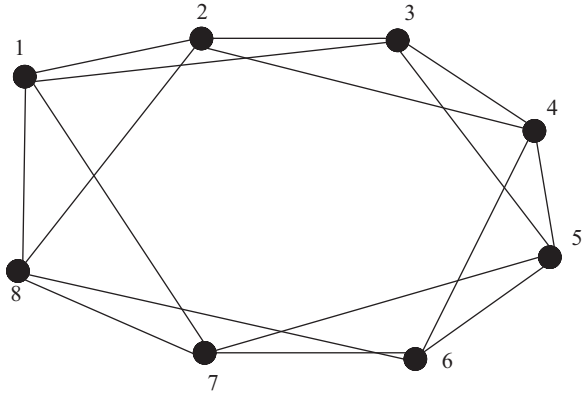


**Fig. 3.** Ring topology with 5 neighbors.

In this variant the velocity equation is given by:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 rand_1(pbest_{ij} - x_{ij}(t)) \quad (19)$$

The social factor of velocities is not used.

- **PSO7. Social-only Particle Swarm Optimization** [29]:

  In this variant the velocity equation is given by:

$$v_{ij}(t+1) = v_{ij}(t) + c_2 rand_2(gbest_j - x_{ij}(t)) \quad (20)$$

  The cognition factor of velocities is not used.

- **PSO8. Local Neighborhood topology Particle Swarm Optimization** [15]:

  The last variant that it is used in this paper is a variant with neighborhood topology for the particles. Neighborhood for a particle is the set of particles with which the particle is connected to. Then, the neighborhood topology of each particle is defined. There are two kinds of population topologies for the Particle Swarm Optimization: the global best (*gbest*) population topology and the local best (*lbest*) population topology [15]. In the *gbest* PSO, the neighborhood for each particle is the entire swarm. The social network employed by the gbest PSO reflects the star topology in which all particles are interconnected (Fig. 1). Thus, the velocities of each particle are updated based on the information obtained from the best particle of the whole swarm. All the previous described variants are based on a global best topology.

  In the *lbest* PSO, each particle has a smaller neighborhood. In this case, the network topology corresponds to the ring topology (Figs. 2 and 3) where each particle communicates with only a limited number of other members of the swarm. The communication is usually achieved with the indices of the particles. Thus, if the size of the neighborhood is equal to three, the selected neighbors for the particle $i$ are the particles $i-1$ and $i+1$. Thus, the velocities of each particle are updated based on the information obtained from the best particle of the neighborhood. The use of particle indices for the creation of the neighborhood is preferred because it is very difficult and computationally expensive to calculate distances between all the particles to find the neighbors of each particle. Furthermore, if the indices are used, then a particle may belong to more than one neighborhood having the possibility of spreading a good solution among different neighborhoods. Usually, the *gbest* PSO converges faster than the *lbest* PSO. On the other hand, the *lbest* PSO has larger diversity in its solutions and, thus, it is more difficult to be trapped in local minima [15].

  In this variant, the equation of velocities is almost the same as in the original algorithm. What changes is the term *gbest* which is replaced by the term *lbest*. The neighborhood may consist of three particles, five particles or more. The equation of velocities is [15]:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 rand_1(pbest_{ij} - x_{ij}(t)) + c_2 rand_2(lbest_{ij} - x_{ij}(t)) \quad (21)$$

  where

$$lbest_{ij} \in \{N_i | f(lbest_{ij}) = min\{f(x_{ij})\}, \forall x \in N_i\} \quad (22)$$

  and the neighbor $N_i$ is defined by [15]:

$$N_i = \{pbest_{i-n_{N_i}}(t), pbest_{i-n_{N_i}+1}(t), \ldots, pbest_{i-1}(t), pbest_i(t), pbest_{i+1}(t), \ldots, pbest_{i+n_{N_i}}(t)\} \quad (23)$$

Thus, for example if the neighbor is equal to 3 and the index of the particle is 5, then, the $lbest_{5j}$ particle will be one of the following particles:

$$pbest_{4j}, \text{pbest}_{\mathbf{5j}}, pbest_{6j}, \quad (24)$$

while if the neighbor is equal to 5, then the $lbest_{5j}$ particle will be one of the following particles:

$$pbest_{3j}, pbest_{4j}, \text{pbest}_{\mathbf{5j}}, pbest_{6j}, pbest_{7j}. \quad (25)$$

### 4.5. Local search

The local search procedure is not applied in the whole swarm but only a number of elitist solutions (particles) are improved even more with the local search algorithms, 2-opt and 3-opt [39]. These algorithms are applied for a number of iterations ($ls_{iter}$) but if for $ls_{iter}/10$ continuous times there is no any improvement to the cost, the local search algorithm is terminated.

The cost is improved even more using the path relinking algorithm. This approach generates new solutions by exploring trajectories that connect high quality solutions. It starts from one of these solutions, called the *starting solution*, and generates a path in the neighborhood space that leads towards the other solution, called the *target solution* [20]. The roles of starting and target solutions can be interchangeable. In the one case, the worst among the two solutions plays the role of the starting solution and the other plays the role of the target solution while in the other case, the roles are changing. There is the possibility the two paths to simultaneously be explored.

In the proposed algorithm, the two solutions are selected as follows:

- For the first $pr_{iter}$ number of iterations, the starting solution is the best solution found so far and the target solution is the second best solution.
- For the iterations from $pr_{iter}$ until the maximum number of iterations, the starting solution is the best solution found so far and the target solution is a solution selected randomly from the whole set of solutions.

During the path relinking procedure, if a better solution from the current best solution is found, then, the current best solution is replaced by this solution.

## 5. Computational results

The whole algorithmic approach was implemented in Matlab R2009a. The algorithm was tested on a set of benchmark instances, the 7 out of 14 benchmark instances proposed by Christofides [10] that have zero service time. 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 51 and 200 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. For the first five instances, nodes are randomly located over a square while for the remaining ones, nodes are distributed in clusters and the depot is not centered. In Table 2, the characteristics of each instance are presented.

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 maximum number of iterations is set to 3500. The $ls_{iter}$ is set equal to 100 and the $pr_{iter}$ is set equal to 560. The minimum and maximum values of $c_1$ and $c_2$ are, respectively, $c_{1,min} = c_{2,min} = 2$ and $c_{1,max} = c_{2,max} = 5$. In every iteration the values of $c_1$ and $c_2$ change based on Eqs. (9) and (10). In the variants of the algorithm that an inertia weight is used, the values of $w_{min}$ and $w_{max}$ are equal to 0.01 and 2, respectively. The size of the neighborhood in the variant that the local best topology is used is set equal to 3. The velocities are initialized with zeros. The deviation of the customers' demand takes the values: $r = 0$, $r = 1$, $r = 2$. The first case ($r = 0$) denotes that there is not any deviation from the actual demand $R$, the second case ($r = 1$) denotes that the deviation is equal to $\pm 1$ while the third case ($r = 2$) denotes that the deviation is equal to $\pm 2$. If for a customer the demand becomes negative, then, this demand takes the value 0.

**Table 2**
Benchmark instances proposed by Christofides [10].

| Instance | $n$ | Capacity | mtl | st |
|---|---|---|---|---|
| C1 | 51 | 160 | $\infty$ | 0 |
| C2 | 76 | 140 | $\infty$ | 0 |
| C3 | 101 | 200 | $\infty$ | 0 |
| C4 | 151 | 200 | $\infty$ | 0 |
| C5 | 200 | 200 | $\infty$ | 0 |
| C11 | 121 | 200 | $\infty$ | 0 |
| C12 | 101 | 200 | $\infty$ | 0 |

A number of tests have been performed using the first variant of the algorithm (PSO1) in order to decide about the number of the particles. In particular, the algorithm is tested for ten times using the first data set (C1) having zero deviation demand ($r = 0$) for 20, 40, 60, 80 and 100 particles ($N$), respectively. Thus, the selected number of particles is the one which gave the smallest average cost. The results of those tests (Cost and number of iterations that the cost was found (Iter)) are presented in Table 3. In this table, the best results, the average results, the best solution, the median, the standard deviation (stdev) and the variance (var) are presented. It is, also, presented the average iteration number where the best solution was found. As it can be seen, the algorithm performs better when the number of particles is equal to 80 or to 100. When the number of particles is equal to 20, 40 or 60, the results are inferior (especially when the number of particles is equal to 20). For the two cases (80 or 100 number of particles) the results are almost equivalent. As we can see when the number of particles is equal to 80, the average solution and the median solution are better than when the number of particles is equal to 100. The standard deviation and the variance are better when the number of particles is equal to 100. This is due to the fact that one run of the algorithm when the number of particles is equal to 80 gave a relatively bad solution (Cost = 535.7765). The number of iterations where the best solution of the algorithm is found is better in the case when the number of particles is equal to 80. Also, in this case only in two runs the solution is greater than 530. Finally, when the number of particles is equal to 80 the algorithm is faster than in the case of 100. For all these reasons a number of particles equal to 80 is selected for the rest of the paper.

Using the selected number of particles, tests are performed for the selection of the variant of the PSO algorithm that performs better. The results of these tests are given in Table 4. In particular, the algorithm is tested for five times using the first data set (C1) having zero deviation demand ($r = 0$), for the classic PSO algorithm and for the seven other variants presented in Section 4.4. The results of these tests (Cost and number of iterations that the cost was found (Iter)) are presented in Table 4. In this table, the best results, the average results, the best solution, the median, the

**Table 3**
Comparisons for the selection of the number of particles.

| C1, $r = 0$ | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $N$ | 20 | | 40 | | 60 | | 80 | | 100 | |
| | Cost | Iter | Cost | Iter | Cost | Iter | Cost | Iter | Cost | Iter |
| | 544.0347 | 504 | 526.6936 | 645 | 537.6127 | 539 | 530.9928 | 546 | 533.257 | 872 |
| | 549.1872 | 907 | 531.7553 | 826 | 528.8605 | 961 | 524.6285 | 937 | 533.257 | 872 |
| | 527.6748 | 766 | 541.1355 | 979 | 524.8098 | 965 | 526.711 | 564 | 527.0322 | 888 |
| | 526.6489 | 629 | 538.6001 | 617 | 536.4079 | 996 | 524.6285 | 621 | 528.1245 | 738 |
| | 542.5711 | 861 | 524.6285 | 585 | 524.9442 | 896 | 524.6285 | 937 | 530.9007 | 960 |
| | 532.2615 | 826 | 539.6085 | 664 | 524.9442 | 745 | 526.711 | 564 | 526.8302 | 783 |
| | 541.4214 | 983 | 529.1147 | 724 | 526.6936 | 772 | 527.6748 | 589 | 532.9958 | 404 |
| | 538.2647 | 661 | 534.2944 | 592 | 536.5233 | 620 | 529.1739 | 659 | 525.1255 | 760 |
| | 529.5191 | 989 | 530.4993 | 870 | 532.2615 | 669 | 524.8098 | 661 | 530.5227 | 784 |
| | 544.0347 | 504 | 528.1419 | 781 | 524.9442 | 795 | 535.7765 | 967 | 524.6111 | 865 |
| Average | 537.56 | 763 | 532.45 | 728.3 | 529.80 | 795.8 | 527.57 | 704.5 | 529.27 | 792.6 |
| Best | 526.6489 | – | 524.6285 | – | 524.8098 | – | 524.6285 | – | 524.6111 | – |
| Median | 539.84 | – | 531.1273 | – | 527.77 | – | 526.711 | – | 529.32 | – |
| stdev | 7.95 | – | 5.73 | – | 5.39 | – | 3.60 | – | 3.35 | – |
| var | 63.24 | – | 32.91 | – | 29.12 | – | 12.96 | – | 11.22 | – |

standard deviation (stdev) and the variance (var) are presented. It is, also, presented the average iteration number where the best solution was found. As it can be seen the algorithm performs better when the PSO3 and PSO7 variants (constriction Particle Swarm Optimization and the social-only model Particle Swarm Optimization, respectively) are used. In four variants (PSO1, PSO2, PSO6 and PSO8) the average cost is larger than 530. In the case of PSO1 and PSO8 this is due to the fact that in one case the results are not so good (543.5613 and 538.3328, respectively). The results in the other four runs are better. The reason that these two variants are considered as unsatisfactory for the solution of the studied problem is that as in 20% of the runs the variants failed in finding a good solution, then, it is possible to fail to find a good solution in the following tests, too. In PSO6 variant, 2 out of 5 runs gave inferior solutions while in PSO2 variant all but one runs gave a solution with cost larger than 530, and, thus, these two variants were not selected for the following tests. The PSO4 and PSO5 variants perform better than the previously mentioned variants. The results of both of them are satisfactory but, in the first variant although the average cost is very low (527.47478), the variance of the solution is not so good, and in the second variant, the average cost is very good (528.88268) but is the worst between the PSO3, PSO4, PSO5, PSO7 variants. These are the reasons that these two variants were not selected for the following tests. The last two variants PSO3 and PSO7 performed very well in all runs. Both of them have low values in average cost, median, standard deviation and variance. Both of them could be selected as the one that it will be used in the following comparisons. The reason that the PSO3 is selected is that

**Table 4**
Comparisons for the selection of the best variant of the algorithm.

| C1, $r = 0$, $N = 80$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Variant | PSO1 | | PSO2 | | PSO3 | | PSO4 | |
| | Cost | Iter | Cost | Iter | Cost | Iter | Cost | Iter |
| | 524.6285 | 937 | 534.9818 | 714 | 527.0094 | 817 | 530.2934 | 790 |
| | 527.1559 | 528 | 532.9958 | 582 | 530.7509 | 987 | 524.8098 | 912 |
| | 531.7553 | 784 | 527.9099 | 844 | 526.8302 | 457 | 533.0137 | 511 |
| | 543.5613 | 955 | 530.2934 | 762 | 524.6111 | 840 | 524.6285 | 612 |
| | 525.1255 | 916 | 530.2629 | 464 | 524.6111 | 721 | 524.6285 | 985 |
| Average | 530.4453 | 824 | 531.28876 | 673.2 | 526.76254 | 764.4 | 527.47478 | 762 |
| Best | 524.6285 | – | 527.9099 | – | 524.6111 | – | 524.6285 | – |
| Median | 527.1559 | – | 530.2934 | – | 526.8302 | – | 524.8098 | – |
| stdev | 7.85 | – | 2.73 | – | 2.51 | – | 3.93 | – |
| var | 61.67 | – | 7.50 | – | 6.30 | – | 15.48 | – |

| C1, $r = 0$, $N = 80$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Variant | PSO5 | | PSO6 | | PSO7 | | PSO8 | |
| | Cost | Iter | Cost | Iter | Cost | Iter | Cost | Iter |
| | 527.242 | 659 | 525.1255 | 617 | 526.9263 | 781 | 538.3328 | 695 |
| | 528.1419 | 566 | 527.0094 | 738 | 530.2934 | 813 | 527.9777 | 584 |
| | 526.6489 | 797 | 538.7111 | 593 | 524.6111 | 791 | 527.9777 | 629 |
| | 532.2448 | 967 | 526.711 | 956 | 526.9263 | 959 | 532.2743 | 938 |
| | 530.1358 | 614 | 533.0907 | 541 | 524.8098 | 860 | 526.8401 | 770 |
| Average | 528.88268 | 720.6 | 530.12954 | 689 | 526.71338 | 840.8 | 530.68052 | 723.2 |
| Best | 526.6489 | – | 525.1255 | – | 524.6111 | – | 526.8401 | – |
| Median | 528.1419 | – | 527.0094 | – | 526.9263 | – | 527.9777 | – |
| stdev | 2.29 | – | 5.67 | – | 2.28 | – | 4.75 | – |
| var | 5.27 | – | 32.21 | – | 5.23 | – | 22.61 | – |

**Table 5**
Comparisons with algorithms from the literature.

| Name | r | BK | GA | | | DE | | | PSO3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Cost | iter | ω | Cost | iter | ω | Cost | iter | Average | ω |
| C1 | 0 | 537.42 | 542.62 | 489 | 0.967 | 537.42 | 967 | 0 | 524.61 | 1314 | 525.93 | -2.383 |
| | 1 | 538.35 | 543.8 | 643 | 1.012 | 538.35 | 944 | 0 | 528.57 | 2874 | 529.90 | −1.816 |
| | 2 | 536.69 | 540.59 | 486 | 0.726 | 536.69 | 649 | 0 | 531.53 | 788 | 533.15 | -0.960 |
| C2 | 0 | 862.39 | 867.86 | 1158 | 0.634 | 862.39 | 978 | 0 | 842.67 | 3493 | 846.73 | -2.286 |
| | 1 | 862.72 | 878.45 | 2364 | 1.823 | 862.72 | 1007 | 0 | 835.71 | 3482 | 849.05 | −3.13 |
| | 2 | 865.86 | 886.87 | 1194 | 2.426 | 865.86 | 719 | 0 | 846.78 | 3484 | 854.4 | -2.203 |
| C3 | 0 | 839.4 | 850.66 | 2269 | 1.341 | 839.4 | 2657 | 0 | 839.39 | 3479 | 846.47 | -0.001 |
| | 1 | 854.51 | 854.51 | 2401 | 0 | 866.15 | 1542 | 1.362 | 843.35 | 3184 | 850.48 | -1.306 |
| | 2 | 851.52 | 854.18 | 1658 | 0.312 | 851.52 | 2083 | 0 | 842.45 | 3317 | 854.04 | -1.065 |
| C4 | 0 | 1095.18 | 1137.24 | 2518 | 3.84 | 1095.18 | 2621 | 0 | 1086.7 | 3439 | 1114.32 | −0.77 |
| | 1 | 1115.37 | 1128.21 | 2314 | 1.15 | 1115.37 | 2614 | 0 | 1108.1 | 3499 | 1119.4 | −0.65 |
| | 2 | 1139 | 1141.18 | 1897 | 0.19 | 1139 | 2518 | 0 | 1117 | 3290 | 1133.86 | −1.93 |
| C5 | 0 | 1495.28 | 1511.21 | 1973 | 1.06 | 1495.28 | 2973 | 0 | 1480.7 | 3460 | 1491.2 | -0.97 |
| | 1 | 1499.17 | 1505.37 | 1489 | 0.41 | 1499.17 | 2146 | 0 | 1483.4 | 3499 | 1500.42 | −1.05 |
| | 2 | 1489.5 | 1497.84 | 2541 | 0.55 | 1489.5 | 2369 | 0 | 1487.4 | 3481 | 1508.92 | −0.14 |
| C11 | 0 | 1055.87 | 1061.47 | 2358 | 0.53 | 1055.87 | 2489 | 0 | 1054.1 | 3476 | 1080.3 | −0.16 |
| | 1 | 1085.49 | 1085.49 | 2417 | 0 | 1088.74 | 2351 | 0.29 | 1075.3 | 3306 | 1111.78 | −0.93 |
| | 2 | 1098.7 | 1117.11 | 2318 | 1.67 | 1098.7 | 2461 | 0 | 1095.3 | 3500 | 1154.9 | −0.31 |
| C12 | 0 | 823.47 | 835.18 | 2519 | 1.42 | 823.47 | 2147 | 0 | 819.55 | 3007 | 821.48 | −0.47 |
| | 1 | 859.79 | 862.49 | 2624 | 0.31 | 859.79 | 2587 | 0 | 856.67 | 3235 | 861.92 | −0.36 |
| | 2 | 861.28 | 864.46 | 2713 | 0.36 | 861.28 | 2614 | 0 | 858.85 | 3375 | 868.86 | −0.28 |

the algorithm found twice the best solution and in all runs the algorithm found the best solution in less iterations, thus the PSO3 needs less computational time than the variant PSO7.

The PSO3 variant was tested for all instances mentioned in Table 2 using the three different values of $r$. The results of these tests for the seven instances are presented in Table 5. The results are, also, compared with the results of two evolutionary algorithms, a Genetic Algorithm and a Differential Evolution algorithm. **Genetic Algorithms (GAs)** [15] are search procedures based on the mechanics of natural selection and natural genetics. A GA is a stochastic iterative procedure that maintains the population size constant in each iteration, called a generation. Their basic operation is the mating of two solutions in order to form a new solution. To form a new population, a binary operator called crossover, and a unary operator, called mutation are used. **Differential Evolution (DE)** is a stochastic, population-based algorithm that was proposed by Storn and Price [54]. DE has the basic characteristics of the evolutionary algorithms as it is an evolutionary algorithm. It focuses in the distance and the direction information of the other solutions. In the differential evolution algorithms [15], initially, a mutation is applied to generate a trial vector and, afterwards, a crossover operator is used to produce one offspring. The mutation step sizes are not sampled from an a priori known probability distribution function as it happens in other evolutionary algorithms but they are influenced by differences between individuals of the current population. The results of these two algorithms have been presented in [41]. In these algorithms, the same formulation of the problem is used. The reason that the algorithm is tested only with these two algorithms is that in many publications in the literature different formulations are used for the Vehicle Routing Problem with Stochastic Demand and, thus, it is not possible to compare the results of the proposed algorithm with the results of these algorithms.

In Table 5, except of the results of the proposed algorithm, the comparisons of the proposed algorithm with the Differential Evolution (DE) algorithm and the Genetic Algorithm (GA) are presented. In the third column of this table, a solution is given that is the best solution derived when comparing for these instances the DE and the GA algorithms (denoted by BK). In the six following columns, the results of the GA and of the DE algorithms are presented, respectively. In the last four columns the results of the proposed algorithm (PSO3) for the three values of demands' deviation for each instance are presented. For each of the algorithms the best cost, the iteration and the quality of the solution ($\omega$) is given. For the proposed algorithm the average results of the five runs are, also, presented. The efficiency of all three algorithms is measured by the quality of the produced solutions. The quality is given in terms of the relative deviation from the best known solution, that is $\omega = (c_{PSO} - c_{BK})/c_{BK}\,\%$, where $c_{PSO}$ denotes the cost of the solution found by PSO algorithm and $c_{BK}$ is the cost of the BK solution. Similarly the quality of the solution of the DE algorithm is given by $\omega = (c_{DE} - c_{BK})/c_{BK}\,\%$, where $c_{DE}$ denotes the cost of the solution found by DE algorithm and the quality of the solution of the GA algorithm is given $\omega = (c_{GA} - c_{BK})/c_{BK}\,\%$, where $c_{GA}$ denotes the cost of the solution found by GA algorithm. As it can be seen from this table, the algorithm gave very good and stable results for all instances. Also, when the number of customers is small, the results of all runs and for all different demands' deviation are close to the average results. Also the algorithm performs better than the other two algorithms in every instance. The improvement of the solution compared to the best known solution is between 0.001 and 3.13 with average improvement equal to 1.105. In the comparison of Differential Evolution algorithm with the Genetic Algorithm, the DE outperforms GA in all but two instances and the improvement of the solution is between 0.191 and 3.698 in the instances that the DE performs better while in the instances that the GA performs better the improvement is 0.299 in the one instance and 1.36 in the other. The improvement of the solution compared to the solution of the GA is between 0.648 and 4.864 with average improvement equal to 2.066 and compared to the DE algorithm is 0.01 to 3.13 with average improvement equal to 1.182.

Although there is a number of papers in the literature that solve the Vehicle Routing Problem with Stochastic Demands, there are no commonly used benchmarks instances and, thus, it is very difficult to compare the approaches between them. Another difficulty that exists for the comparisons of different methods is the fact that most researchers use a different approach to deal with the most important issue of the Vehicle Routing Problem with Stochastic Demands, the Route Failure. Mainly, there are two approaches to

**Table 6**
Results in the second set of benchmark instances.

| | $n$ | Capacity | First approach | | | Second approach |
|---|---|---|---|---|---|---|
| | | | BK | Christiansen and Lysgaard [9] | Goodson et al. [23] | PSO |
| A-n32-k5 | 32 | 100 | 853.6 | 853.6 | 853.6 | 821.65 |
| A-n33-k5 | 33 | 100 | 704.2 | 704.2 | 704.2 | 687.04 |
| A-n33-k6 | 33 | 100 | 793.9 | 793.9 | 793.9 | 769.62 |
| A-n34-k5 | 34 | 100 | 826.87 | 827.87 | 826.87 | 789.88 |
| A-n36-k5 | 36 | 100 | 858.71 | – | 858.71 | 836.05 |
| A-n37-k5 | 37 | 100 | 708.34 | 708.34 | 708.34 | 693.18 |
| A-n37-k6 | 37 | 100 | 1030.73 | 1030.75 | 1030.73 | 999.72 |
| A-n38-k5 | 38 | 100 | 775.14 | 778.09 | 775.14 | 756.56 |
| A-n39-k5 | 39 | 100 | 869.18 | 869.18 | 869.18 | 853.08 |
| A-n39-k6 | 39 | 100 | 876.6 | 876.6 | 876.6 | 847.92 |
| A-n44-k6 | 44 | 100 | 1025.48 | 1025.48 | 1025.48 | 978.83 |
| A-n45-k6 | 45 | 100 | 1026.73 | – | 1026.73 | 997.41 |
| A-n45-k7 | 45 | 100 | 1264.83 | 1264.83 | 1264.99 | 1175.45 |
| A-n46-k7 | 46 | 100 | 1002.22 | 1002.41 | 1002.22 | 984.98 |
| A-n48-k7 | 48 | 100 | 1187.14 | – | 1187.14 | 1132.15 |
| A-n53-k7 | 53 | 100 | 1124.27 | – | 1124.27 | 1096.6 |
| A-n54-k7 | 54 | 100 | 1287.07 | – | 1287.07 | 1223.23 |
| A-n55-k9 | 55 | 100 | 1179.11 | – | 1179.11 | 1124.3 |
| A-n60-k9 | 60 | 100 | 1529.82 | – | 1529.82 | 1454.15 |
| E-n22-k4 | 22 | 6000 | 411.57 | 411.57 | 411.57 | 390.99 |
| E-n33-k4 | 33 | 8000 | 850.27 | 850.27 | 850.27 | 847.38 |
| E-n51-k5 | 51 | 160 | 552.26 | – | 552.26 | 544.86 |
| P-n16-k8 | 16 | 35 | 512.82 | 512.82 | 512.82 | 455.21 |
| P-n19-k2 | 19 | 160 | 224.06 | 224.06 | 224.06 | 213.51 |
| P-n20-k2 | 20 | 160 | 233.05 | 233.05 | 233.05 | 226.79 |
| P-n21-k2 | 21 | 160 | 218.96 | 218.96 | 218.96 | 218.13 |
| P-n22-k2 | 22 | 160 | 231.26 | 231.26 | 231.26 | 229.45 |
| P-n22-k8 | 22 | 3000 | 681.06 | 681.06 | 681.06 | 590.72 |
| P-n23-k8 | 23 | 40 | 619.52 | 619.52 | 619.53 | 536.34 |
| P-n40-k5 | 40 | 140 | 472.5 | 472.5 | 472.5 | 471.24 |
| P-n45-k5 | 45 | 150 | 533.52 | – | 533.52 | 530.52 |
| P-n50-k10 | 50 | 100 | 760.94 | – | 760.94 | 739.51 |
| P-n50-k7 | 50 | 150 | 582.37 | – | 582.37 | 570.94 |
| P-n50-k8 | 50 | 120 | 669.81 | – | 669.81 | 659.19 |
| P-n51-k10 | 51 | 80 | 809.7 | 809.7 | 812.74 | 795.43 |
| P-n55-k10 | 55 | 115 | 745.7 | – | 745.7 | 737.87 |
| P-n55-k15 | 55 | 70 | 1068.05 | 1068.05 | 1068.05 | 1008.6 |
| P-n55-k7 | 55 | 170 | 588.56 | – | 588.56 | 587.95 |
| P-n60-k10 | 60 | 120 | 804.24 | – | 804.24 | 772.86 |
| P-n60-k15 | 60 | 80 | 1085.49 | 1085.49 | 1087.41 | 1021.58 |

solve this issue. Both of them calculate the expected length of an a priori route or set of routes beginning and ending at the depot, such that each customer is assigned to exactly one route and the expected demand assigned to each route does not exceed vehicle capacity and the goal is to minimize the expected cost. The differences of these approaches are the way they treat the route failure. In the first one [9,23,34], vehicles are required to continue on their assigned routes until a route failure occurs, in which case capacity is replenished at the depot and, then, the vehicle returns at the location of the customer where the route failure occurs and continues the service. In the second one, which is the one that is used in this paper, there is a preventive restocking strategy [4,61] where in order to avoid a route failure, the optimum choice is the choice of a threshold value where if the residual load after a customer is greater or equal to the threshold value, then, it is better to move to the next customer, otherwise it is better to return to the depot. Another difference of these two approaches is that in the first one a set of vehicles can be used but in our approach (the second one) the way that we cope with the route failure leads to use only one vehicle.

Taking into consideration the different approaches of solving the Vehicle Routing Problem with Stochastic Demands, it is very difficult to find a common set of benchmark instances. In this paper, we use a set of benchmark instances based on the classic Vehicle Routing Problem instances proposed by Christofides [10]. In [9,23], another set of benchmark instances is given [62]. Thus, it will be very interesting to also apply our method to these instances in order to test the effectiveness of the PSO algorithm. It is very important to mention that the results of the proposed algorithm will not be directly comparable with the results of the other two researches due to the different way that we solve the issue of the route failure. As it was mentioned previously in [9,23], when a route failure occurs, the vehicle performs a direct route to the depot and, then, it returns back to the location of the route failure and, thus, an additional cost is added to the total cost of the solution. Using the preventive restocking policy in this paper, the vehicle never returns to the same customer as this policy prohibits the occurrence of a route failure. For the Christofides benchmark instances, the transformation of the customer demands from a deterministic demand to stochastic variables with known probability, the approach that was described in Section 4.3 is used. However, for the other set of benchmark instances as it is desired to follow the same transformation approach as the one proposed in [9,23], we assume that customer demands are independent Poisson random variables with the mean demand for each customer equal to the deterministic value of the demand given in the corresponding VRP problem.

In Table 6, the results of the proposed algorithm in the second set of benchmark instances are presented. The algorithm was tested in forty instances, the same with the ones used in [9,23], with number of nodes from 16 to 60. In the three first columns of Table 6, the name of the instance, the number of nodes and the capacity of the vehicles are presented. Columns 4–6 present the

results from the literature based on the first approach of dealing with the issue of the route failure occurrence, more precisely, the Best Known (BK) solutions in column 4, the results of the Christiansen and Lysgaard [9] (column 5) and the results of Goodson et al. [23] (column 6), respectively. In the last column the results of the proposed algorithm based on the second approach of dealing with the issue of the route failure occurrence is presented. As it is mentioned previously the results of these two approaches are not directly comparable between them. The reason that we gave these results is that they could be used as a basis for testing and comparisons in future researches using either the one approach for dealing with route failure or the other. It is very important to notice that both approaches are very effective in solving the Vehicle Routing Problem with Stochastic Demands and the significant improvement of the results in the proposed method is only because of the different treatment of the route failure and, not that the other two algorithms are inferior of the proposed one.

## 6. Conclusions and future research

In this paper a new algorithm, based on the Particle Swarm Optimization, for the solution of the Vehicle Routing Problem with Stochastic Demands is presented. This algorithm is a combination of the Particle Swarm Optimization algorithm with the 2-opt and 3-opt local search algorithms and with the path relinking strategy. As a number of different variants of the Particle Swarm Optimization algorithm have been published, mainly with a different equation of velocities, we have tested the most known of them and we found the one that performs better for the VRPSD. This method was the constriction Particle Swarm Optimization (PSO3) combined with the local search strategies mentioned previously and it was chosen for all comparisons. Another issue that we have to dealt with was the fact that the PSO algorithm is suitable for continuous optimization problems. Thus, it was a challenge to find an effective transformation of the solutions of PSO in discrete values without loosing information from this procedure. As there are a number of formulations in the literature for the Vehicle Routing Problem with Stochastic Demands we have chosen the one that fits better to the problem that we would like to solve. In the Vehicle Routing Problem with Stochastic Demands there is no data set as in the Capacitated Vehicle Routing Problem or the Vehicle Routing Problem with Time Windows where all researchers have tested their algorithms. Thus, we have tested the proposed algorithm in some of the benchmark instances proposed by Christofides for the Capacitated Vehicle Routing Problem adding different demands' deviation. In order to see the effectiveness of the proposed algorithm we compared the results with the results of other algorithms from the literature for the same instances. The proposed algorithm found new best solution for the Vehicle Routing Problem with Stochastic Demands in all instances. Our future research will be focused in two different directions. The one direction will be towards solving the VRP with Stochastic Demands using other nature inspired techniques like Clonal Selection Algorithm, Honey Bees Mating Optimization etc., while the other direction will be towards solving with the proposed algorithm even more complicated problems like Vehicle Routing Problem with Stochastic Demands and Time Windows or the Dynamic Vehicle Routing Problem.

## References


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

[2] A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications, Natural Computing 7 (2008) 109–124.

[3] R.W. Bent, P. Van Hentenryck, Scenario-based planning for partially dynamic vehicle routing with stochastic customers, Operations Research 52 (6) (2004) 977–987.

[4] L. Bianchi, M. Birattari, M. Manfrin, M. Mastrolilli, L. Paquete, O. Rossi-Doria, T. Schiavinotto, Hybrid metaheuristics for the vehicle routing problem with stochastic demands, Journal of Mathematical Modelling and Algorithms 5 (1) (2006) 91–110.

[5] L. Bianchi, M. Dorigo, L.M. Gambardella, W.J. Gutjahr, A survey on metaheuristics for stochastic combinatorial optimization, Natural Computing 8 (2) (2009) 239–287.

[6] L. Bodin, B. Golden, Classification in vehicle routing and scheduling, Networks 11 (1981) 97–108.

[7] L. Bodin, B. Golden, A. Assad, M. Ball, The state of the art in the routing and scheduling of vehicles and crews, Computers and Operations Research 10 (1983) 63–212.

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

[10] N. Christofides, A. Mingozzi, P. Toth, The vehicle routing problem, in: N. Christofides, A. Mingozzi, P. Toth, C. Sandi (Eds.), Combinatorial Optimization, Wiley, Chichester, 1979, pp. 315–338.

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

[12] M. Clerc, J. Kennedy, The particle swarm: explosion, stability and convergence in a multi-dimensional complex space, IEEE Transactions on Evolutionary Computation 6 (2002) 58–73.

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

[14] M. Dror, G. Laporte, F.V. Louveaux, Vehicle routing with stochastic demands and restricted failures, ZOR – Methods and Models of Operations Research 37 (1993) 273–283.

[15] A.P. Engelbrecht, Computational Intelligence: An Introduction, 2nd ed., John Wiley and Sons, England, 2007.

[16] M.L. Fisher, 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, vol. 8, North Holland, Amsterdam, 1995, pp. 1–33.

[17] M. Gendreau, G. Laport, R. Seguin, Stochastic vehicle routing, European Journal of Operational Research 88 (1996) 3–12.

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

[19] M. Gendreau, G. Laporte, J.Y. Potvin, 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.

[20] F. Glover, M. Laguna, R. Marti, Scatter search and path relinking: advances and applications, in: F. Glover, G.A. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, Boston, 2003, pp. 1–36.

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

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

[23] J.C. Goodson, J.W. Ohlmann, B.W. Thomas, Cyclic-order neighborhoods with application to the vehicle routing problem with stochastic demand, European Journal of Operational Research 217 (2012) 312–323.

[24] Z.G. Guo, K.L. Mac, A heuristic algorithm for the stochastic vehicle routing problems with soft time windows, Congress on Evolutionary Computation 2004 2 (2004) 1449–1456.

[25] D. Haugland, S.C. Ho, G. Laporte, Designing delivery districts for the vehicle routing problem with stochastic demands, European Journal of Operational Research 180 (2007) 997–1010.

[26] L.M. Hvattum, A. Lïkketangen, G. Laporte, A heuristic solution method to a stochastic vehicle routing problem, in: Proceedings of TRISTAN V-The Fifth Triennial Symposium on Transportation Analysis, 2004.

[27] A. Juan, J. Faulin, S. Grasman, D. Riera, J. Marull, C. Mendez, Using safety stocks and simulation to solve the vehicle routing problem with stochastic demands, Transportation Research Part C 19 (2011) 751–765.

[28] A.A. Juan, J. Faulin, J. Jorba, J. Caceres, J.M. Marques, Using parallel and distributed computing for real-time solving of vehicle routing problems with stochastic demands, Annals in Operations Research (2012), 10.1007/s10479-011-0918-z.

[29] J. Kennedy, The particle swarm: social adaptation of knowledge, in: Proceedings of the IEEE International Conference on Evolutionary Computation, 1997, pp. 303–308.

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

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

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

[33] A.S. Kenyon, D.P. Morton, Stochastic vehicle routing with random travel times, Transportation Science 37 (2003) 69–82.

[34] G. Laporte, F. Semet, 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.

[35] G. Laporte, M. Gendreau, J.Y. Potvin, F. Semet, Classical and modern heuristics for the vehicle routing problem, International Transactions on Operations Research 7 (2000) 285–300.
[36] H. Lei, G. Laporte, B. Guo, The capacitated vehicle routing problem with stochastic demands and time window, Computers and Operations Research 38 (2011) 1775–1783.
[37] X. Li, P. Tian, S.C.H. Leung, Vehicle routing problems with time windows and stochastic travel and service times: models and algorithm, International Journal of Production Economics 125 (2010) 137–145.
[38] D. Lichtblau, Discrete optimization using mathematica, in: N. Callaos, T. Ebisuzaki, B. Starr, J.M. Abe, D. Lichtblau (Eds.), in: World Multi-conference on Systemics, Cybernetics and Informatics (SCI 2002), Vol. 16, International Institute of Informatics and Systemics, 2002, pp. 169–174.
[39] S. Lin, Computer solutions of the traveling salesman problem, Bell Systems Technical Journal 44 (1965) 2245–2269.
[40] Y. Marinakis, A. Migdalas, 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.
[41] Y. Marinakis, M. Marinaki, P. Spanou. A memetic differential evolution algorithm for vehicle routing problem with stochastic demands and customers, submitted for publication.
[42] J.E. Mendoza, B. Castaniera, C. Guereta, A.L. Medagliab, N. Velascob, A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands, Computers and Operations Research 37 (2010) 1886–1898.
[43] I. Minis, A. Tatarakis, Stochastic single vehicle routing problem with delivery and pick up and a predefined customer sequence, European Journal of Operational Research 213 (2011) 37–51.
[45] C. Novoa, R. Storer, An approximate dynamic programming approach for the vehicle routing problem with stochastic demands, European Journal of Operational Research 196 (2009) 509–515.
[46] F.B. Pereira, J. Tavares, Bio-inspired Algorithms for the Vehicle Routing Problem, Studies in Computational Intelligence, vol. 161, Springer, Berlin Heidelberg, 2008.
[47] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization. An overview, Swarm Intelligence 1 (2007) 33–57.
[48] M. Protonotarios, G. Mourkousis, I. Vyridis, T. Varvarigou, Very large scale vehicle routing with time windows and stochastic demand using genetic algorithms with parallel fitness evaluation, in: HPCN 2000, LNCS 1823, 2000, pp. 467–476.
[49] M. Reimann, Analyzing a vehicle routing problem with stochastic demands using ant colony optimization, in: A. Jaszkiewicz, M. Kaczmarek, J. Zak, M. Kubiak (Eds.), Advanced OR and AI Methods in Transportation, Publishing House of Poznan University of Technology, 2005, pp. 764–769.
[50] N. Secomandi, Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands, Computers and Operations Research 27 (2000) 1201–1225.
[51] Z. Shen, M. Dessouky, F. Ordonez, The Stochastic Vehicle Routing Problem for Large-scale Emergencies. Technical Report 2005–2006, Department of Industrial and Systems Engineering, University of Southern California, 2005.
[52] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Proceedings of 1998 IEEE World Congress on Computational Intelligence, 1998, pp. 69–73.
[53] W.R. Stewart, B.L. Golden, Stochastic vehicle routing: a comprehensive approach, European Journal of Operational Research 14 (1983) 371–385.
[54] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359.
[55] C.D. Tarantilis, Solving the vehicle routing problem with adaptive memory programming methodology, Computers and Operations Research 32 (2005) 2309–2327.
[56] K.C. Tan, C.Y. Cheong, C.K. Goh, Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation, European Journal of Operational Research 177 (2007) 813–839.
[58] F. Tillman, The multiple terminal delivery problem with probabilistic demands, Transportation Science 3 (1969) 192–204.
[59] P. Toth, D. Vigo, The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications, Siam, 2002.
[60] S. Yan, C.J. Chi, C.H. Tang, Inter-city bus routing and timetable setting under stochastic demands, Transportation Research, Part A 40 (2006) 572–586.
[61] W.H. Yang, K. Mathur, R.H. Ballou, Stochastic vehicle routing problem with restocking, Transportation Science 34 (2000) 99–112.
[62] http://www.coin-or.org/SYMPHONY/branchandcut/VRP/data/Vrp-All.tgz