MODELING THE MULTI-COMPARTMENT VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS

Jan Melechovský[(a)]

[(a)]Department of Econometrics, University of Economics, náměstí W. Churchilla 4, 130 67 Prague 3, Czech Republic

[(a)]jan.melechovsky@vse.cz

ABSTRACT
Vehicle routing problems represent a class of combinatorial optimization problems widely studied in the literature. Practical applications often involve a level of uncertainty, which complicates the decision making process. This paper presents two models of the multi-compartment vehicle routing problem with stochastic demands (MCVRPSD). The problem consists in finding a minimum cost set of vehicle routes serving a set of customers. Each customer can require the delivery of *m* products. The products must be transported in different compartments due to their physical incompatibility. The problem is modeled as a stochastic program with recourse. The recourse action consists in a return to the depot vertex whenever the servicing vehicle cannot satisfy the customer's demand for a particular product. In an alternative model the failure can also occur due to time constraints. A hybridized evolutionary algorithm is presented to address the problem.

Keywords: vehicle routing, stochastic programming, metaheuristic, multiple compartments

1. INTRODUCTION

The multi-compartment vehicle routing problem (MCVRP) extends the classical formulation of the vehicle routing problem (VRP). In the VRP, a set of customer nodes is given. Each customer requires the delivery of a nonnegative quantity of a product. A vehicle fleet situated at a depot node is available to service the customers. In the classical formulation of the VRP, the vehicle fleet is unlimited and homogeneous, i.e. all vehicles have equal capacity. The solution of the VRP consists in determining a minimum cost set of vehicle routes satisfying the total demand of each customer. Each route is originating and ending at the depot node and respects the capacity of a vehicle. The VRP was first formulated by Dantzig and Ramser (1959). A survey on VRP applications and solution approaches can be found in e.g. Toth and Vigo (2002) or Golden et al. (2008). The VRP is a NP-hard problem since it generalizes the well-known travelling salesman problem. Exact algorithms are limited to instances of a moderate size. Hence metaheuristic algorithms have been proposed to address VRPs of a larger size.

In the MCVRP, each customer can require the delivery of multiple products. The products have specific characteristics and cannot be transported together in one room. Each vehicle is therefore equipped with multiple separated compartments, each dedicated to transport a particular product. In the basic formulation of the MCVRP the vehicle fleet is again assumed to be unlimited and homogeneous. The objective is to satisfy all customer demands at minimum cost. Each product must be transported in the right vehicle compartment and the capacity of each compartment must be respected. Each vehicle route starts and ends at the depot. In the MCVRP, the delivery of two different products can be done in two separate routes. This feature is called partial split of the delivery.

A typical application of the MCVRP is e.g. the delivery of petroleum products to petrol stations using tank trucks with several compartments (see e.g. Avella et al. (2004), Brown and Graves (1981), Brown et al. (1987). In this application, the demand of a customer for a particular petrol type is quite large and often saturates the whole compartment capacity. The main problem then consists of determining the assignment of requests to tank trucks, such that the available capacity is used optimally. The routing part of the problem is then relatively easy since a TSP with a small number of stops is solved for each truck separately. Another practical application is the delivery of groceries to convenience stores Chajakis and Guignard (2003). Each product requires different temperature, e.g. low and ambient temperature compartments must be used. Customers have limited inventory capacities and small but frequent orders are therefore preferred. Animal food distribution to farms is an example of the MCVRP addressed by El Fallahi et al. (2008). The authors modeled the case when the food must be transported separately for certain types of animals due to sanitary reasons. The MCVRP can also model systems of waste co-collection when different types of waste can be collected using vehicles with multiple compartments Muyldermans and Pang (2010).

Practical applications of routing problems often involve a level of uncertainty in the input data. Such a situation arises when the demand of customer is rather a random variable than a deterministic value. The

MCVRP with stochastic demands (MCVRPSD) models this issue. For each customer, the demand for a given product is a random variable with a known probability distribution characterized with a mean and a deviation. The real demand is always nonnegative and is less than or equal the vehicle capacity Q_p. However, the exact demand is known only upon vehicle's arrival to the customer.

The basic modeling approaches to stochastic routing problems can be classified into static or dynamic models. Routing decisions in dynamic models are performed in multiple stages and depend on prior demands realizations. A re-optimization action is taken at each stage depending on the remaining vehicle capacity and set of unserved customers. An example of a dynamic model is the Markov Decision Process proposed by Secomandi and Margot (2009).

In the static model, routing decisions remain unchanged. This approach is more convenient for difficult-to-solve problems since the problem is solved only once. A typical static stochastic routing model consists of two stages. In the first stage an *a priori* solution is calculated while in the stage the routes are executed. In case of a route failure, e.g. the vehicle does not have enough quantity of a particular product to satisfy the total demand of a customer, a recourse action is performed. The recourse action may e.g. consist in a return trip to the depot, refill of the vehicle compartments up to their capacity, and a resume trip back to the customer node where the failure occurred. A two stage stochastic programming model for the MCVRPSD was proposed by Mendoza et al. (2010).

This paper presents a two-stage stochastic programming model for the MCVRPSD. A solution based on the expected demand quantities is determined in the first stage. The presented model of the problem considers two particular situations. In the first case the stochastic demand affects only the available quantity of a particular product to be delivered. The solution must be adapted so that all customers receive their demanded quantities. In the alternative model, the factor of time is additionally taken into account. The total time duration of a route is limited to T_{max} while the unloading time s_i at customer i is a function of the unloaded quantity.

2. PROBLEM MODEL

The problem is defined on undirected graph $G(V,E)$, where $V=\{0,1,2,\dots,n\text{-}1\}$ denotes the set of n nodes and E denotes the set of edges. The subset $N = V \setminus \{0\}$ represents the customers indexed from 1 to n-1 while the index 0 is reserved for the depot node. Each customer $i \in N$ can require the delivery of a nonnegative quantity q_{ip} of product $p \in P$. If $q_{ip} = 0$ the product is not ordered by the customer. The number of products is denoted m in the sequel. In the stochastic model, the demand of customer i for product p is a random variable $\lambda_{i,p}$ with a known probabilistic distribution with mean $\mu_{i,p}$ and standard deviation $\sigma_{i,p}$. The exact value of q_{ip} is therefore known only upon vehicle's arrival to i. A service time $s_{ip} = \alpha \cdot q_{ip}$ is associated with the delivery of product p to customer i. α is a constant value representing the time needed to unload one unit of a product. A fleet K of identical vehicles based in the depot is available to service the customers. Each vehicle is equipped with m compartments. Each compartment has a fixed capacity Q_p to accommodate product p. Each edge $e(i,j) \in E$ is associated with travel cost c_{ij} and travel time t_{ij}. Both values are assumed to be nonnegative and both sets of values satisfy the triangle inequality. The solution of the MCVRP consists in determining a set of vehicle routes, each route starting and ending at the depot, such that the demands of each customer are totally satisfied, the capacities of each vehicle are respected and the total travel cost is minimized. The problem is NP-hard since it extends the classical VRP.

The two stage stochastic programming model for the MCVRPDS was proposed in Mendoza et al. (2010). Let S be the planned (a priori) solution consisting of n_s routes. Each route $r \in S$ starts and ends at the depot and visits a sequence of customers $(v_1, v_2, \dots, v_l)$. The total cost of the solution in the second stage is given by the equation:

$$\boldsymbol{C(S) = \sum_{r \in S} C_r + \sum_{r \in S} F_r(\vec{\lambda})} \qquad (1)$$

where C_r is the planned cost of route r and $F_r(\vec{\lambda})$ is the cost of route failure (recourse).

Then the problem to solve in the first stage is to determine the set of routes S minimizing the expected cost:

$$\boldsymbol{E[C(S)] = \sum_{r \in S} C_r + \sum_{r \in S} E[F_r(\vec{\lambda})]} \qquad (2)$$

The maximum route duration constraints can be expressed as follows:

$$\boldsymbol{T_r + E[G_r(\vec{\lambda})] + E[H_r(\vec{\lambda})] \le T_{max} \;\forall\, r \in S} \qquad (3)$$

where T_r denotes the travel time of the planned route r, $G_r(\vec{\lambda})$ the time of the recourse trips and $H_r(\vec{\lambda})$ the expected unloading time that is counted for r.

The cost of the recourse depends on the failure probability $P_r(v_i)$ at customer v_i and the cost c_{0i}:

$$\boldsymbol{E[F_r(\vec{\lambda})] = \sum_{i=1}^{l} 2c_{0v_i} \times P_r(v_i)} \qquad (4)$$

Analogically, the time of the recourse can be calculated as:

$$E[G_r(\vec{\lambda})] = \sum_{i=1}^{l} 2t_{0v_i} \times P_r(v_i) \qquad (5)$$

For the calculation of the failure probability $P_r(v_i)$ at node v_i, Mendoza et al. (2010) proposed the following approximation scheme:

$$P_r(v_i) = \sum_{j=0}^{i-1} \left[\prod_{p=1}^{m} \Phi(v_{j+1}, v_{i-1}, p) - \prod_{p=1}^{m} \Phi(v_{j+1}, v_i, p) \right] \cdot P_r(v_j) \qquad (6)$$

The term $\Phi(v_{j+1}, v_{i-1}, p)$ expresses the cumulative probability that the demand for product p between nodes at positions $j+1$ and $i-1$ does not exceed the capacity Q_p of the corresponding vehicle compartment. Hence the first product approximates the probability that the routes remains feasible after servicing customers $v_{j+1},\ldots,v_{i-1}$. Similarly, the second product approximates the probability that the route remains feasible after servicing customer v_i.

Finally, the expected unloading time depends on the unloaded quantity at each node:

$$E[H_r(\vec{\lambda})] = \sum_{i=1}^{l} \sum_{p=1}^{m} \alpha\lambda_{v_i,p} \qquad (7)$$

3. THE MULTI-START EVOLUTIONARY ALGORITHM

The evolutionary local search method (ELS) was originally proposed by Merz and Wolf (2007) for a peer-to-peer problem in telecommunications. It extends the classical iterated local search (ILS). A detailed description of the ILS can be found in Lourenço et al. (2003). The ILS, given an incumbent solution s, successively generates child solutions using a perturbation mechanism and a local search. At each iteration, s is updated with a child solution only if some acceptance criterion is met. The perturbation mechanism serves as a diversification tool while the local search intensifies the search in the current solution neighborhood. The ELS additionally generates multiple child solutions at one iteration. Only the best child solution is kept and it becomes the incumbent solution s only if it fulfills the acceptance criterion.

The multi-start feature extends the capability of ELS. It can be viewed as the greedy randomized adaptive search procedure (GRASP) in which the local search is implemented as the ELS. In the MS-ELS the search is restarted with a randomized initial solution each time when the algorithm is being trapped in the local optimum. The original idea of multi-start ELS (MS-ELS) is due to Prins (2009).

```
1: $tc \leftarrow 0$
2: $ni \leftarrow 0$
3: **repeat**
4:   **if** $tc = 0$ **then** $\rho \leftarrow 1$
5:   **else** $\rho \leftarrow \rho_{max}$
6:   $s \leftarrow$ Randomized_Initial_Heuristic($\rho$)
7:   **if** $tc = 0$ **then** $s^* \leftarrow s$
8:   $\pi \leftarrow \pi_{min}$
9:   **for** $N_e$ iterations **do**
10:    $cs^* \leftarrow \emptyset$
11:    $Cost(cs^*) \leftarrow \infty$
12:    **for** $N_c$ child solution iterations **do**
13:      $cs \leftarrow s$
14:      $cs' \leftarrow$ Perturbation($cs, \pi$)
15:      $cs'' \leftarrow$ Local_Search($cs'$)
16:      **if** $Cost(cs'') < Cost(cs^*)$ **then**
17:        $cs^* \leftarrow cs''$
18:      **endif**
19:      $tc \leftarrow tc + 1$
20:      **if** $Cost(cs^*) < Cost(s^*)$ **then**
21:        $ni \leftarrow 0$
22:        $s^* \leftarrow cs''$
23:      **else** $ni \leftarrow ni + 1$
24:      **if** $tc \geq N_t$ **or** $ni \geq NI_t$ **goto** line 31
25:    **endfor**
26:    **if** $Cost(cs^*) < Cost(s)$ **then**
27:      $\pi \leftarrow \pi_{min}$
28:      $s \leftarrow cs^*$
29:    **else** $\pi \leftarrow \min(\pi + 1, \pi_{max})$
30:  **endfor**
31: **until** $tc \geq N_t$ **or** $ni \geq NI_t$
```

Algorithm 1. The Multi-Start Evolutioary Local Search

The MS-ELS is used to solve the first stage problem minimizing the expected cost $E[C(S)]$. The algorithm implementation is shown in Algorithm 1. First the total iterations counter tc and the non-improving iterations counter ni are initialized with 0. Multiple restarts of the algorithm with a different incumbent solution s are ensured by the main loop between lines 3 – 31. The solution diversity within each restart is ensured with a randomized initial heuristic using a randomization parameter ρ denoting the number of admissible extensions of a partial solution at an iteration of the initial heuristic. The first initial solution is obtained with no randomization (ρ = 1). The best solution s^* is initialized at the beginning of the first iteration with the initial solution. The perturbation parameter π of the ELS is set to its minimum value π_{min}. This parameter controls the perturbation strength and it is dynamically updated during the run. The ELS (lines 9 – 30) performs at most N_e iterations. At every ELS iteration, N_c child solutions are generated using the perturbation mechanism and the local search. Only the best child solution *cs** among all child solutions generated is kept in the memory. In case it improves also the global best solution *s** the latter is updated with *cs** and *ni* is set to 0. The algorithm can be terminated prematurely if one of the iterations counter attains its maximum limit. If *cs** improves the incumbent solution *s*, it is updated with *cs** and π is reset to π_{min}. Otherwise π is incremented. The algorithm stops when either *tc* reaches the number of maximum total iterations N_t or *ni* reaches the number of maximum non-improving iterations NI_t.

The key components of the MS-ELS are the randomized initial heuristic, perturbation mechanism and a local search mechanism. The following subsections provide implementation details of each component.

3.1. Initial heuristic

The initial solution is obtained with a simple construction heuristic. The solution is initialized with an empty route. Then all customers with some unsatisfied demand are scanned and the best insertion of a customer into a route is determined. If the customer cannot be inserted into any already existing route, the insertion into a new route is considered instead. The criterion for the insertion is the least increase of travel costs. The procedure terminates when the demand of all customers is fully satisfied. The method is randomized with a parameter ρ denoting the number of best insertions (in terms of insertion cost) determined in each iteration which are recorded into a list. The insertion which is finally performed is selected randomly from that list.

3.2. Perturbation

The perturbation mechanism is one of the two routines of ELS that modify the incumbent solution. It plays the role of a diversification tool in the general ELS framework since it performs several random operations on a solution. Thus it can be interpreted as a mutation operator used in genetic algorithms. The procedure is controlled with a parameter π denoting the number of operations to be performed. In our implementation the operation is a removal and relocation of a customer. Given a route *r* in the solution, π customer nodes are randomly selected and removed from *r*.

Each of the π removed customers is then tested for a feasible insertion into some of the remaining routes of the solution. If such feasible insertion is detected, the customer is relocated to its new position. If not, new route visiting only this single customer is added to the solution.

3.3. Local search

Local search is applied to the solution modified by the perturbation mechanism. Its purpose is to improve the solution using a set of operators. It intensifies the search and the improved solution represents a local optimum within the given solution neighborhood. Together with the perturbation mechanism it enables the algorithm to explore effectively the solution space and find potentially good solutions.

The implemented local search procedure relies on three operators:

a) *2-opt* – replaces two arcs and reorders nodes,
b) *Path Exchange* – interchanges two sub-paths between two routes,
c) *Relocate* – relocates a sequence of nodes within the same route or between two routes,
d) *Swap* – swaps two sequences of nodes between two routes.

Moreover, *Relocate* and *Swap* can operate with sequences of *k* consecutive nodes in a route instead of one single customer. The parameter *k* defines the size of the neighborhood to be explored. Starting with *k = 1*, the size is dynamically increased up to a maximum possible value k_{max} if no improvement was found with the current value of *k*.

The operators are illustrated in Figures 1 – 6. Each operator replaces a certain number of arcs in the solution. The cancelled arcs are depicted with dashed lines. *2-opt* in Figure 1 operates on a single route. It replaces a pair of arcs and reorders the intermediate sequence of customers. *Path Exchange* in Figure 2 is a variation of 2-*opt* but it operates on a pair of routes. It replaces one arc in each route and the order of customers remains preserved. *Relocate* can operate on a single route (Figure 3) or on a pair of routes (Figure 4). This operator replaces three arcs. Finally, *Swap* replaces four arcs either within a single route or a pair of routes. The operator is depicted in Figure 5 and Figure 6.

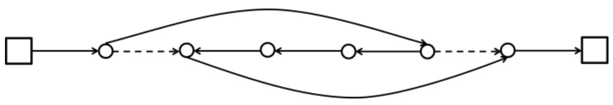
Figure 1. 2-opt exchange.

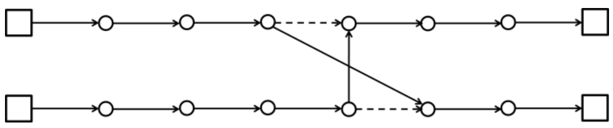
Figure 2. Path exchange.

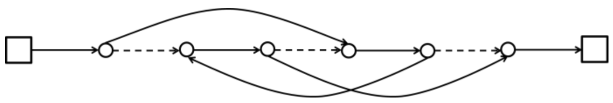
Figure 3. Relocate on a single route.

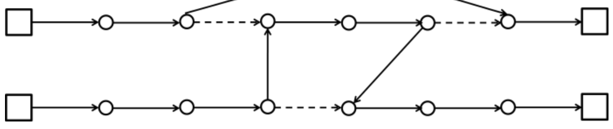
Figure 4. Relocate on a pair of routes.

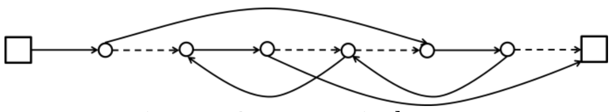

Figure 5. Swap on a single route.

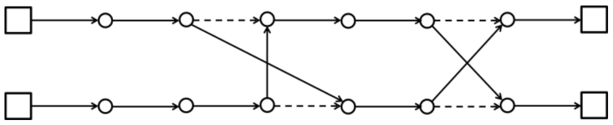

Figure 6. Swap on a pair of routes.

The operators are applied sequentially in the above given order with the first improvement strategy – i.e. the first detected improving move is performed. The search stops when no improving move can be found by any operator. Each move must be checked for the feasibility. This involves the time windows as well as the capacity. The time windows feasibility can be checked in *O(1)* if an information of the maximum feasible shift of each node visit is kept in the memory. However, the update of this information must be done for each node and requires *O(n)*. The capacity must be checked for all vehicle compartments if the move involves two routes. It can be implemented in *O(1)*.

Every insertion or removal of a customer requires the evaluation of the cost difference generated by the operation. The cost difference is composed of the part given by the deterministic travel cost and the part expressing the impact of the operation on the failure probability. The second part requires the recalculation for each customer subsequent the operation. Even the approximation scheme for the evaluation of the failure probability given by Equation 6 is computationally expensive. As shown in Mendoza et al. (2010), the recalculation of the failure probability requires $O(mn^2)$. For this reason the authors suggested to evaluate the recourse cost only in some cases. In the presented algorithm, the recourse cost is evaluated only when selecting the best insertions within the initial heuristic and the perturbation. The local search procedure relies only on the deterministic travel cost.

4. COMPUTATIONAL EXPERIMENTS

4.1. Testing environment

Two sets of testing instances for deterministic MCVRP were proposed by El Fallahi et al. (2008). The authors derived two data sets from standard VRP instances available at Beasley's *OR Library* (http://people.brunel.ac.uk/~mastjjb/jeb/). Each dataset is based on 20 instances. The first 14 instances (vrp1 – vrp14) contain 50 – 199 customers. The last six instances contain 76 – 484 customers. In both datasets the number of products is 2. The first dataset (*Set 1*) is derived in a straightforward way: the capacity of compartments and customer demands is divided into two equal parts. In the second dataset (*Set 2*), the demand q_{i1} for product 1 is generated randomly for each i. The demand for the second product is defined as $q_{i2} = q_i - q_{i1}$, where q_i is the original demand of customer i. The capacity Q_1 of the first compartment is determined as follows from the average demand $\bar{Q}_1$ for the first product, the average demand $\bar{Q}_2$ for the second product, and the vehicle capacity Q in the original VRP: $Q_1 = (Q \times \bar{Q}_1)/(\bar{Q}_1 + \bar{Q}_2)$. The capacity of the second compartment is given by $Q_2 = (Q \times \bar{Q}_2)/(\bar{Q}_1 + \bar{Q}_2)$.

Mendoza et al. (2010) adapted the instances for the stochastic MCVRP. The demands follow normal distribution with $\mu_{i,p} = d_{i,p}$ for each $i \in N$ and $p \in P$. The standard deviation is calculated as $\sigma_{i,p} = 0.3 \times \mu_{i,p}$. Some instances involve the distance upper limit implicitly. For the others the limit is $L = 4 \times \max_{i \in N} c_{0i}$. These instances do not involve the service time. For the purpose of preliminary tests, *Set 3* has been derived from *Set 2*. The service time is defined as $s_{ip} = \alpha \cdot q_{ip}$, where $\alpha = 1$. The travel time is set equal to the travel cost ($t_{ij} = c_{ij}\ \forall e(i,j) \in E$) and the maximum time of a route is defined as $T_{max} = L + \alpha \cdot n_r \cdot (\bar{Q}_1 + \bar{Q}_2)$. n_r denotes the maximum number of customers serviced in one route in the best solution of the original instance in *Set 2*.

4.2. Parameters setting

The proposed algorithm requires eight parameters to be set up. The parameters are summarized in Table 1. The determination of an appropriate set of parameters is not trivial and the task constitutes a particular part of computational experiments. Several tests of the MS-ELS have been therefore carried out on the 14 MCVRP instances from *Set 1*. In these preliminary experiments the demand was considered to be deterministic. Each instance was solved with 41 different parameter configurations and each test was executed five times with various rand seed settings.

Table 1: List of algorithm parameters

Notation	*Description*
N_t	Maximum number of total iterations
NI_t	Maximum number of non-improving iterations
N_e	Number of ELS iterations
N_c	Number of child solutions
π_{min}	Minimum value of perturbation parameter
π_{max}	Maximum value of perturbation parameter
ρ_{max}	Randomization parameter
k_{max}	Maximum size of a sequence in local search (*Relocate* or *Swap*)

The results are shown in Table 2. Columns 2 – 9 present the configuration values. The last two columns present the average gap to the best known solutions (BKS) of the concerned instances and the average running time in seconds respectively. The results are sorted in non-decreasing order of the average gap (ranging from 1.9 % to 5.3 %). The selected configuration is highlighted in bold.

Table 2: Results of MS-ELS obtained with different parameter configurations

	N_t	NI_t	N_e	N_c	π_{min}	π_{max}	ρ_{max}	k_{max}	**Gap**	**cpu**
1	1000	800	10	15	1	5	3	3	1.9	313.8
2	1000	800	10	15	1	3	3	3	2.8	283.2
3	1000	800	10	10	1	5	3	3	2.9	296.1
4	1000	800	10	10	1	3	3	3	2.9	291.1
5	500	500	10	10	1	3	3	5	3.0	381.4
6	1000	500	10	15	1	3	3	3	3.0	227.3
7	1000	500	10	10	1	5	3	3	3.0	236.2
8	1000	500	10	15	1	5	3	3	3.0	260.3
9	**800**	**500**	**10**	**15**	**1**	**5**	**3**	**3**	**3.0**	**215.2**
10	800	500	10	15	1	3	3	3	3.0	225.8
11	800	500	10	10	1	5	3	3	3.1	218.2
12	1000	500	10	10	1	3	3	3	3.1	257.2
13	500	200	10	10	1	3	3	5	3.3	252.4
14	1000	300	10	10	1	5	3	3	3.3	160.5
15	1000	300	10	15	1	5	3	3	3.3	162.2
16	500	500	10	10	1	5	3	3	3.3	158.9
17	500	500	10	15	1	5	3	3	3.3	152.2
18	800	500	10	10	1	3	3	3	3.3	227.0
19	1000	300	10	15	1	3	3	3	3.3	150.3
20	500	500	10	10	1	3	3	4	3.3	252.9
21	500	500	10	15	1	3	3	3	3.4	145
22	500	200	10	10	1	3	3	4	3.6	157.6
23	500	500	10	10	1	3	3	2	3.7	80.5
24	500	200	10	15	1	3	3	3	3.7	101.5
25	800	200	10	15	1	3	3	3	3.7	107.2
26	800	200	10	15	1	5	3	3	3.7	106.6
27	500	200	10	15	1	5	3	3	3.7	102.3
28	200	200	10	10	1	3	3	5	3.8	153.8
29	800	200	10	10	1	5	3	3	3.8	110.3
30	500	200	10	10	1	5	3	3	3.8	106.9
31	1000	300	10	10	1	3	3	3	3.9	161.0
32	500	500	10	10	1	3	3	3	3.9	146.1
33	200	200	10	10	1	3	3	4	4.1	101.6
34	200	200	10	15	1	5	3	3	4.3	59.3
35	200	200	10	15	1	3	3	3	4.3	56.9
36	500	200	10	10	1	3	3	3	4.3	95.2
37	800	200	10	10	1	3	3	3	4.3	104.1
38	500	200	10	10	1	3	3	2	4.4	51.0
39	200	200	10	10	1	5	3	3	4.6	61.9
40	200	200	10	10	1	3	3	3	4.9	59.8
41	200	200	10	10	1	3	3	2	5.3	33.1

The selection of parameter values was done with respect to the expected impact on algorithm's performance. Most attention was therefore given to parameters N_t and NI_t. Contrarily, N_e , π_{min} and ρ_{max} were always set to the same values. N_e was fixed to 10 and the reason was that it's impact is strongly correlated with N_t and N_c . Hence varying these values provided similar behavior as varying the values of N_e . Regarding π_{min}, the performance did not change or it was even worse for slightly larger values than 1. The randomization parameter ρ_{max} was set to 3, since larger values led to initial solutions of poor quality. N_c alternated between 10 and 15. Lower values than 10 led to worse results while values significantly larger (about 20 or more) did not bring any positive impact. Moreover, it would reduce the number of restarts since the portion of total iterations run on the same starting solution would increase. The variability of tested values of π_{max} and k_{max} was also quite moderate for similar reasons. The parameter π_{max} was tested only with values 3 and 5 and k_{max} was tested with values 3, 4, and 5. Larger values of π_{max} led to equal or worse solutions while larger values of k_{max} increased the computing time significantly.

4.3. Computational experiments

Preliminary computational experiments of the stochastic version of the proposed MS-ELS were conducted on instances vrp1 – vrp14 of *Set 2* and *Set 3*. As in Mendoza et al. (2010), the results were obtained with a single of the MS-ELS and the stochastic simulation involved 50 000 scenarios. The algorithm was coded in C++ and compiled with Embarcadero RAD Studio 2010. The tools provided by the Boost library were used for statistical calculations and simulations.

The results obtained for *Set 2* are summarized in Table 3. The table compares the performance of the MS-ELS to the best known solutions including the estimated cost of recourse over 50 000 replications reported in Mendoza et al. (2010). Gap is the percentage difference between the solution obtained with MS-ELS and the BKS. Computing times in the last column of the table are in seconds. The solutions found by our MS-ELS are generally close to the BKSs. The gap ranges from 0.2 % to 3.5 %. Mendoza et al. (2010) reported the results for two versions of the memetic algorithm. The basic version much slower but it calculated most of the best known solutions. A faster version of their memetic algorithm represents uses a spare capacity strategy, which relies on the computation of the deterministic problem and leaving some spare capacity in each vehicle compartment. This modification of the algorithm does not require the costly computations of the recourse probability and hence it is far more effective compared to the stochastic version. In Table 3, our results are compared to the stochastic version of the memetic algorithm.

Table 3: Results of the MS-ELS on the adapted Set 2 instances

Instance	*n*	*BKS*	*MS-ELS*	*Gap*	*Time*
vrpnc1	50	568.7	569.9	0.2	123.3
vrpnc2	75	976.3	978.3	0.2	198.4
vrpnc3	100	891.8	894.6	0.3	645.3.
vrpnc4	150	1187.3	1198.2	0.9	1230.5
vrpnc5	199	1497.3	1509.5	0.8	2056.7
vrpnc6	50	577.6	586.0	1.5	98.5
vrpnc7	75	1000.9	1020.4	1.9	103.6
vrpnc8	100	922.1	931.3	1.0	289.5
vrpnc9	150	1255.8	1273.2	1.4	432.6
vrpnc10	199	1572.5	1587.3	0.9	1027.1
vrpnc11	120	1259.1	1302.6	3.5	1356.4
vrpnc12	100	999.1	1011.4	1.2	823.6
vrpnc13	120	1558.8	1564.2	0.3	567.2
vrpnc14	100	1027.4	1036.8	0.9	431.8

The second set of computational experiments was conducted on *Set 3*. The instances in this set involve the service time which is spent by the vehicle in the customer site. Recall that the service is a function of the unloaded quantity. The results are reported in Table 4. BKS is the best solution value obtained during five executions of the algorithm. The column MS-ELS reports the average value over the five runs. The gap ranges from 0.8 % to 4.7 %, which can be considered as an acceptable variance in results obtained in different runs of the algorithm.

Table 4: Results of the MS-ELS on the adapted Set 3 instances

Instance	*n*	*BKS*	*MS-ELS*	*Gap*	*Time*
vrpnc1	50	577.1	605.5	0.8	142.9
vrpnc2	75	1066.7	1075.8	3.9	231.0
vrpnc3	100	932.2	937.0	4.1	684.6
vrpnc4	150	1189.5	1241.8	3.2	729.4
vrpnc5	199	1642.7	1648.9	2.1	1988.5
vrpnc6	50	628.1	630.7	2.9	102.2
vrpnc7	75	1037.7	1051.1	3.1	61.4
vrpnc8	100	1000.2	1015.1	4.7	401.3
vrpnc9	150	1355.3	1417.0	3.6	537.1
vrpnc10	199	1599.0	1673.1	0.9	1020.4
vrpnc11	120	1357.4	1367.6	3.4	495.8
vrpnc12	100	1053.0	1097.6	1.8	787.2
vrpnc13	120	1620.5	1680.8	1.2	463.4
vrpnc14	100	1079.9	1091.4	0.9	394.8

The experiments on *Set 3* are rather preliminary. Detailed analysis of the impact of a stochastic service time will be subject of further research. The time limit T_{max} in *Set 3* was defined such that the optimal solutions do not differ too much from *Set 2*. The solution values obtained on *Set 3* are slightly above the solutions of *Set 2*.

CONCLUSIONS

This paper presents a MS-ELS algorithm for the MCVRP with stochastic demands. The solution approach is based on a two-stage stochastic programming formulation proposed in Mendoza et al. (2010). The proposed metaheuristic calculates the estimate of solution total cost when the demand for products is a random variable. The algorithm was tested on benchmarks from the literature. Preliminary results indicate that the MS-ELS is a competitive algorithm compared notably to the memetic algorithm proposed by Mendoza et al. (2010). An extension of the problem formulation presented in the paper involves variable service time depending on the unloaded quantity. This property influences the feasibility of the vehicle route due to the limit of time imposed. The next logical step in the problem model will be the inclusion of customer time windows into the model. The variable service time will then influence the feasibility of arrivals to customer sites.

ACKNOWLEDGMENTS

The research was fully supported by the Czech Science Foundation, project reg. number P402/12/P635. This financial support is greatly acknowledged.

REFERENCES

Avella P., Boccia M., and Sforza A., 2004. Solving a fuel delivery problem by heuristic and exact approaches. *European Journal of Operational Research*, 152 (1), 170 – 179.

Brown G. G. and Graves G. W., 1981. Real-time dispatch of petroleum tank trucks. *Management Science*, 27 (1), 19 – 32.

Brown G. G., Ellis C. J., Graves G. W., and Ronen D. 1987. Real-time, wide area dispatch of mobil tank trucks. *Interfaces*, 17 (1), 107 – 120.

Chajakis E. D. and Guignard M., 2003. Scheduling deliveries in vehicles with multiple compartments. *Journal of Global Optimization*, 26, 43 – 78.

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

El Fallahi A., Prins C., and Wolfler Calvo R., 2008 . A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Computers & Operations Research*, 35 (5), 1725 – 1741.

Golden B. L., Raghavan S. , and Wasil E. A., 2008. editors. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer.

Lourenço H., Martin O., and Stützle T., 2003. Iterated local search. In Frederick S. Hillier, Fred Glover, and Gary Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 320–353. Springer New York.

Mendoza J. E., Castanier B., Guéret C., Medaglia A. L., and Velasco N., 2010. A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. *Computers & Operations Research*, 37 (11), 1886 – 1898.

Merz P. and Wolf S., 2007. Evolutionary local search for the super-peer selection problem and the p-hub median problem. In T. Bartz-Beielstein et al., editors, *Lecture notes in computer science*, volume 4771, pages 1 – 15. Springer Berlin/Heidelberg.

Muyldermans L. and Pang G., 2010. On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm. *European Journal of Operational Research*, 206 (1), 93 – 103.

Prins C., 2009. A grasp x evolutionary local search hybrid for the vehicle routing problem. In F.B. Pereira and J. Tavares, editors, *Bio-inspired algorithms for the vehicle routing problem*, volume 16, pages 35 – 53. Springer Berlin/Heidelberg.

Secomandi N. and Margot F., 2009. Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Operations Research*, 57 (1), 214–230.

Toth P. and Vigo D., editors, 2002. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

AUTHORS BIOGRAPHY

Jan Melechovsky is a researcher at University of Economics, Prague. He got his Ph.D. degree in Technology and Management of Transportation Systems from Czech Technical University in Prague in 2009. His research activities are focused to optimization problems in transportation networks.