A Variable Neighborhood Search for the Generalized Vehicle Routing Problem with Stochastic Demands

Benjamin Biesinger, Bin Hu, and Günther Raidl

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Favoritenstraße 9–11/1861, 1040 Vienna, Austria
`{biesinger|hu|raidl}@ads.tuwien.ac.at`

Abstract. In this work we consider the generalized vehicle routing problem with stochastic demands (GVRPSD) being a combination of the generalized vehicle routing problem, in which the nodes are partitioned into clusters, and the vehicle routing problem with stochastic demands, where the exact demands of the nodes are not known beforehand. It is an NP-hard problem for which we propose a variable neighborhood search (VNS) approach to minimize the expected tour length through all clusters. We use a permutation encoding for the cluster sequence and consider the preventive restocking strategy where the vehicle restocks before it potentially runs out of goods. The exact solution evaluation is based on dynamic programming and is very time-consuming. Therefore we propose a multi-level evaluation scheme to significantly reduce the time needed for solution evaluations. Two different algorithms for finding an initial solution and three well-known neighborhood structures for permutations are used within the VNS. Results show that the multi-level evaluation scheme is able to drastically reduce the overall run-time of the algorithm and that it is essential to be able to tackle larger instances. A comparison to an exact approach shows that the VNS is able to find an optimal or near-optimal solution in much shorter time.

Keywords: generalized vehicle routing problem, stochastic vehicle routing problem, variable neighborhood search, stochastic optimization

1 Introduction

The generalized vehicle routing problem with stochastic demands (GVRPSD) combines the vehicle routing problem with stochastic demands (VRPSD) with the generalized vehicle routing problem (GVRP) and is a stochastic combinatorial optimization problem.

In the GVRPSD we are given a weighted complete undirected graph $G = (V, E)$ with a set of nodes V and a set of edges E. The edges $(i, l) \in E$ are

This work is supported by the Austrian Science Fund (FWF) grant P24660-N23

weighted with distances $d_{il} \geq 0$. The set of nodes is partitioned into m disjoint subsets or clusters $C = \{C_0, C_1, \dots, C_m\}, C_0, \dots, C_m \subseteq V$, such that $C_0 \cup C_1 \cup \dots \cup C_m = V$. Node $v_0 \in V$ is a dedicated depot and the only node of cluster C_0. Each other cluster $C_j, \forall j = 1, \dots, m$ has an associated demand ξ_j which is a random variable following a known discrete probability distribution, i.e., we know for each cluster C_j the probability $p_{jk} = P(\xi_j = k)$ that cluster j has a demand of $k \geq 0$. Furthermore, we are given a vehicle with a limited capacity Q. For avoiding the necessity of multiple visits we assume that $p_{jk} = 0,\ \forall j = 1, \dots, m,\ \forall k > Q$. The aim is to find a route visiting exactly one node from each cluster $C_1, \dots, C_m$ exactly once and thereby distributing goods according to the clusters' actual demands. The current load of the vehicle decreases each time a cluster demand is satisfied but is refilled to Q each time it returns to the depot. The amount of how much the load gets decreased by visiting cluster j is dependent on the actual realization of ξ_j, which becomes known only upon arrival. Possibly, the vehicle will get empty, i.e., the current load becomes zero and the vehicle has to restock at the depot before continuing the route. Such an event is called a *stockout*. A common approach for solving such stochastic optimization problems is the use of a-priori routes, which has already been used for several probabilistic problems [3, 13, 4]. A-priori tours are planned before the actual realizations of the random variables are known but taking their probability distributions into account. The aim of the problem is to minimize the expected length of the tour under all possible a-posteriori tours.

In the literature there are several restocking strategies described for the VRPSD which can be adapted to the GVRPSD as well. The by far most common is the *standard restocking* policy [15, 7, 12, 20] where on each stockout the vehicle returns to the depot, refills its load, and continues its tour at the last visited node. Another method for handling stockouts is *re-optimization* [22]: whenever a stockout occurs the vehicle returns to the depot and then the tour through the remaining clusters is re-planned. Apart from the re-optimization approach, which can be problematic when implemented in practical applications, the *preventive restocking* policy [24, 4, 16, 17] is the most cost efficient.

In the preventive restocking method return trips to the depot can also be performed before an actual stockout occurs. It originates from the observation that a repeated visit of the same node after a restocking is usually expensive and can often be avoided if the restocking is done after servicing the preceding cluster. Especially on instances where the triangle inequalities hold, a restocking from the preceding cluster is always more cost efficient. Another advantage from using the preventive restocking strategy is that it is sufficient to plan one giant tour through all clusters when the problem is not further constrained. Yang et al. [24] proved this property for the VRPSD and it can be directly applied to the GVRPSD as well. Along with that proof they also proposed an evaluation procedure for the giant tour representation, which we will discuss in Section 3.

Like the GVRP this problem can be applied to the field of healthcare logistics, in which medical supplies are delivered to districts and the distributing company does not know beforehand how much supply is needed. If it does not matter to which hospital in each district these supplies are delivered this problem can be modelled as a GVRPSD. Another application domain is urban waste management, where refuse collecting vehicles gather waste from districts returning it to a central landfill site and the total amount of waste is not known beforehand.

In this work we describe a variable neighborhood search (VNS) [10] approach for the GVRPSD with preventive restocking. We use the giant tour representation and an evaluation procedure similar to the one used for the VRPSD [24], which is based on dynamic programming (DP). In addition to this solution evaluation procedure we also propose a multi-level evaluation scheme which iteratively approximates the quality of a solution candidate until it can be discarded as being inferior or the exact objective is obtained. In the next section the related work for this problem is discussed. Section 3 is dedicated to the solution representation and the associated evaluation procedure including the multi-level evaluation scheme. The actual variable neighborhood search and its operators are described in Section 4. We present computational results for this approach in Section 5 and draw conclusion of our work along with thoughts on future work in Section 6.

2 Related Work

To the best of our knowledge the GVRPSD has not been considered in the literature so far. However, there are many related problems which have been broadly discussed like the VRPSD, which is a special case of the GVRPSD but each cluster is a singleton. Especially the work by Yang et al. [24] and Bianchi et al. [4], which both used the preventive restocking strategy, has been inspiring to the approach for the GVRPSD presented here. Yang et al. [24] showed that planning multiple tours cannot lead to a better solution in the VRPSD. They also presented an evaluation procedure based on dynamic programming for the giant tour representation. The same evaluation is also used by Bianchi et al. [4] who developed several metaheuristics and an approximate delta evaluation for the VRPSD. The most recent work for the VRPSD utilizing the preventive restocking policy is by Marinakis et al. [17] who also applied the same evaluation procedure and presented a clonal selection algorithm. They reported promising results on their benchmark set and compared their algorithm to two versions of a particle swarm optimization [16], to a differential evolution, and a genetic algorithm.

Another related problem is the GVRP, which is the special case of the GVRPSD with deterministic demands. There are usually capacity or distance constraints on the used vehicles and therefore several routes have to be devised. There are several exact and heuristic solution approaches in the literature [2, 19, 18, 8, 1]. Since we are considering a giant tour approach the generalized traveling salesman problem (GTSP) is also closely related. The GTSP was originally introduced independently in [11], [23], and [21]. In Fischetti et al. [5, 6] integer linear programming models are formulated and analyzed. For our construction heuristic we use one of their formulations.

3 Solution Representation and Evaluation

In the introduction we mentioned that it is sufficient to plan one giant tour through all clusters and therefore we use a solution representation based on the sequence of clusters that are visited in the tour. From this permutation of clusters we compute the expected cost using a DP algorithm based on the DP for the VRPSD [24]. We adapt it to the GVRPSD by also considering that we only have to visit one node per cluster. The worst case run-time complexity remains $\mathcal{O}(|V|Q^2)$.

First we describe the notations used in the DP. The function used for the recursion $f_{ij}(q)$ is defined for all $q = 0, \dots, Q$, $j = 0, \dots, m$, $i = 0, \dots, |C_j|$ and can be interpreted as the remaining cost of the tour after servicing the i-th node of cluster j with the residual vehicle capacity q. We also define an auxiliary function $b_j(l)$ which returns the l-th node of cluster j. Let us assume that the clusters of the tour we want to evaluate are relabeled such that the tour is $t = (C_0, C_1, \dots, C_m, C_0)$. Then the DP recursion is given by:

$$f_{ij}(q) = \min\{f^p_{ij}(q), f^r_{ij}(q)\}$$

$$\begin{aligned} f^p_{ij}(q) = \min_{l=0,\dots,|C_{j+1}|}\{d_{b_j(i),b_{j+1}(l)} \quad &+ \sum_{k=0}^{q} f_{l,j+1}(q-k)p_{j+1,k} \\ &+ \sum_{k=q+1}^{Q} 2d_{b_{j+1}(l),0} + f_{l,j+1}(q+Q-k)p_{j+1,k}\} \\ f^r_{ij}(q) = d_{b_j(i),0} + \min_{l=0,\dots,|C_{j+1}|}\{d_{0,b_{j+1}(l)} \quad &+ \sum_{k=0}^{Q} f_{l,j+1}(Q-k)p_{j+1,k}\} \end{aligned}$$

$$\forall q = 0, \dots, Q, j = 0, \dots, m, i = 0, \dots, |C_j|$$

and the boundary condition

$$f_{im}(q) = d_{b_m(i),0}, \quad \forall q = 0, \dots, Q,\ i = 0, \dots, |C_m|$$

The basic principle of this recursion is that for each node i and each vehicle load q it is computed if it is more cost-efficient to proceed directly to the next cluster which costs $f^p_{ij}(q)$ or to make a preventive restock which costs $f^r_{ij}(q)$. The total expected cost of the tour t is then given by the value of $f_{0,0}(Q)$. For our VNS such an expensive solution evalution is inconvenient for larger instances with a large vehicle capacity. In the next section we describe a method to potentially reduce the run-time of the solution evaluation within the VNS framework, which can also be applied to other metaheuristics.

3.1 Multi-Level Evaluation Scheme

In this section we describe a multi-level evaluation scheme (ML-ES) to iteratively estimate the exact objective value of a solution candidate with increasing accuracy until we either know that it cannot be better than the best solution found so far or we know its exact value. The basic idea is to scale down both the vehicle capacity and the probability distribution of the demand for each cluster accordingly. Since the time needed for the solution evaluation is quadratically dependent on Q, a large performance gain in terms of run-time is expected when Q is decreased.

In our ML-ES there are $\log_2 Q$ levels of approximation, where level 0 is the exact evaluation and $\log_2 Q$ is the roughest approximation level. Starting with level 0, increasing the level by one means to scale down the vehicle capacity Q and all demand distributions p_{jk} by a factor of two. We introduce a new vehicle capacity Q^i and new probabilities p^i_{jk} subject to level i, which are defined in the following way:

$$Q^0 = Q \tag{1}$$

$$p^0_{jk} = p_{jk} \qquad \forall j = 1, \dots, |C|,\ k = 0, \dots, Q \tag{2}$$

$$Q^i = \left\lceil \frac{Q^{i-1}}{2} \right\rceil \qquad \forall i = 1, \dots, \lceil \log_2 Q \rceil \tag{3}$$

$$p^i_{jk} = p^{i-1}_{j,2k} + p^{i-1}_{j,2k+1} \qquad \forall j = 1, \dots, |C|,\ k = 0, \dots, Q^i, \quad \forall i = 1, \dots, \lceil \log_2 Q \rceil \tag{4}$$

Figure 3.1 shows exemplarily for one cluster how the probability distribution for the demand changes at each level.

Not only is level $i \geq 1$ an approximation, but its objective value is also a lower bound for the objective value of the preceding level $i-1$, which we will show next.

Lemma 1. *With increasing level i the ratio of the scaled expected demand of each cluster to the vehicle capacity Q^i is non-increasing.*

Proof. We have to show for each cluster j and each demand $0 \leq k \leq Q$ that

$$\frac{\sum_{k=0}^{Q^i} k p^i_{jk}}{Q^i} \leq \frac{\sum_{k=0}^{Q^{i-1}} k p^{i-1}_{jk}}{Q^{i-1}}, \quad \forall i = 1, \dots, \lceil \log_2 Q \rceil$$

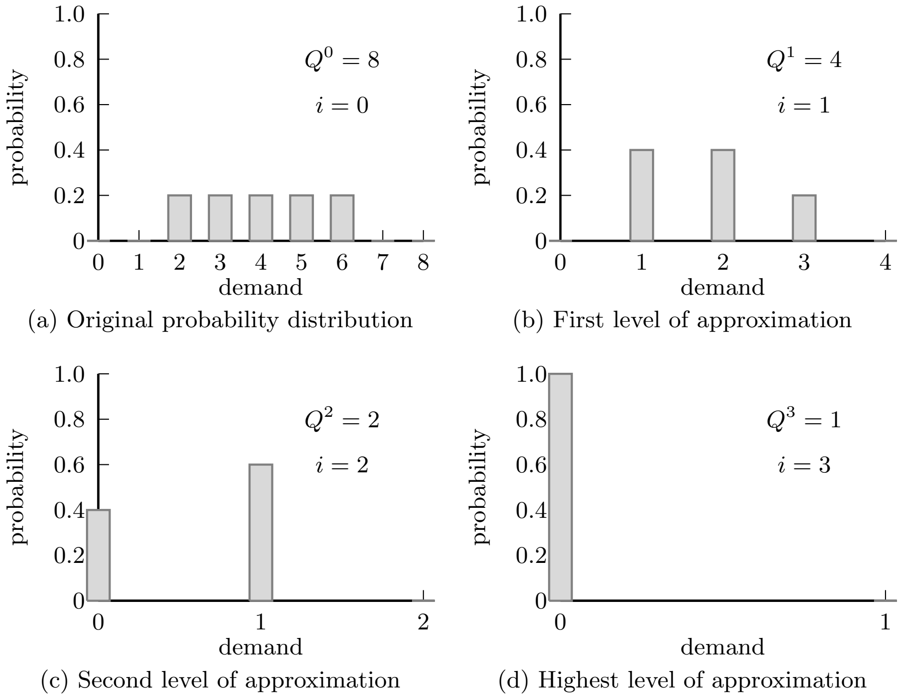
is valid. Suppose to the contrary that for one cluster j and one demand k the following holds:

$$\frac{k p^i_{jk}}{Q^i} = \frac{k(p^{i-1}_{j,2k} + p^{i-1}_{j,2k+1})}{\frac{Q^{i-1}}{2}} > \frac{2k p^{i-1}_{j,2k} + (2k+1) p^{i-1}_{j,2k+1}}{Q^{i-1}} = \frac{k p^{i-1}_{jk}}{Q^{i-1}}$$

$$2k p^{i-1}_{j,2k} + 2k p^{i-1}_{j,2k+1} > 2k p^{i-1}_{j,2k} + 2k p^{i-1}_{j,2k+1} + p^{i-1}_{j,2k+1}$$

$$0 > p^{i-1}_{j,2k+1}$$

Obviously, this is a contradiction because all probabilities must be non-negative. Therefore, as no $\frac{k p^i_{jk}}{Q^i}$ can be larger than $\frac{k p^{i-1}_{jk}}{Q^{i-1}}$ for any cluster j this also holds for the sum over all demands, which proves the Lemma. □



(a) Original probability distribution (b) First level of approximation (c) Second level of approximation (d) Highest level of approximation

Fig. 1. An exemplary demand probability distribution and its different levels of approximation.

Theorem 1. *Let $c^i(t)$ be the objective value of a tour t on approximation level i. For each tour t it holds that $c^i(t) \leq c^{i-1}(t), \forall i = 1, \ldots, \lceil \log_2 Q \rceil$.*

Proof. Due to Lemma 1 it follows that the total expected relative demand of all clusters on level i is smaller or equal to that of level $i - 1$. So we can possibly service more customers before a restocking is needed and therefore the resulting objective $c^i(t)$ value is a lower bound to the exact objective value $c^0(t)$ and to the objective value at the preceding level $c^{i-1}(t)$. □

Algorithm 1 describes our ML-ES in pseudocode, where $DP(t, i)$ executes the DP described above with the scaled vehicle capacity and probability distributions according to (1–4). Algorithm 1 returns either the exact objective value of t if $DP(t, 0)$ is executed or a lower bound to the exact value otherwise. In the latter case the solution candidate can immediately be discarded because we know that it cannot be better than the best solution found so far.

4 Variable Neighborhood Search

The proposed VNS follows the general variable neighborhood search scheme as described in [9]. The underlying variable neighborhood descent (VND) considers

Algorithm 1: ML-ES(t, $bestObj$)

```
Input : tour $t$, objective value of best solution found so far $bestObj$
Output: exact or approximate objective value
$obj = 0$;
$i = \lceil \log_2 Q \rceil$;
while $obj < bestObj \wedge i \geq 0$ do
    $obj = DP(t, i)$;
    $i = i - 1$;
return $obj$;
```

three neighborhood structures which are well-known for the TSP: *1-shift*, *2-opt*, and *Or-opt*. As a shaking procedure for diversification we perform k^2 moves in the k-th neighborhood with $k = 1, \ldots, 3$.

4.1 Initial Solution

For finding an initial solution two types of construction heuristics are considered. The first, *farthest insertion*, is well-known for the classical traveling salesman problem and suited for Euclidean instances only. It builds iteratively a tour by starting at the depot cluster and inserting the cluster which is farthest away from the last inserted cluster at the best possible position. For that purpose we have to define distances between clusters, which is done by computing the geometric centers of clusters by taking the average of the x- and y-coordinates of its nodes. Then the distance between two clusters is the Euclidean distance between their centers.

An alternative but much more time-consuming method for finding a starting solution is solving the GTSP relaxation of the problem. From the solution of the GTSP relaxation we extract the cluster sequence which is then our initial solution. The GTSP is solved exactly by using a branch-and-cut algorithm with CPLEX and the E-GTSP formulation described in [6].

4.2 Neighborhood Structures

Three types of neighborhood structures are used in the VND part, which are searched with a best improvement step function in the order they are described here.

1-shift : A cluster is shifted to another position of the tour.

2-opt : A subsequence of the tour is inverted.

Or-opt : First two, then three consecutive clusters are shifted to another position of the tour. Note that Or-opt usually starts by shifting only one cluster in the tour but we covered this case by our first neighborhood structure and omit it here.

5 Computational Results

The VNS is implemented in C++ and for the GTSP starting solution CPLEX in version 12.5 is used. All our tests were carried out on a single core of an Intel Xeon with 2.53 GHz and 3GB RAM. We created Euclidean instances for the GVRPSD[1] based on instances for the GVRP [2] by assigning the original demand values to be the expected demands and deciding independently at random for each cluster if it is a *low spread* or a *high spread* cluster. For low spread clusters the set of possible demands is $\pm 10\%$ of the expected value and for the high spread it is $\pm 30\%$. All of these demand values are considered equally likely, so we assumed a uniform distribution over these values. We do not consider demand values smaller than zero or larger than Q. Due to space limitations the numerical results presented in this section are based on a representative selection of 37 instances out of the originally proposed 158 instances. These instances are selected such that a comparison to an existing exact method is possible. The full result tables can be found at our website[1].

In the following tables the column *Instance* contains the name of the instance, followed by the number of nodes n and the number of clusters m. Additionally the expected number of restocks $E[nr]$ are presented. Then the results for the different configurations are given with their (average) objective values, their standard deviations (sd) if applicable and either the total run-time in seconds $t[s]$ for the deterministic configurations or the average time when the best solution is identified $t^*[s]$.

First we compare the results of the different starting solutions, farthest insertion (FI) and GTSP, with a subsequent VND using the neighborhood structures and the order described in Section 4.2. Table 1 shows the (deterministic) numerical results for these configurations. Additionally it contains a third configuration where the ML-ES is used along with the GTSP starting solution.

The results indicate that both starting solutions produce similarly good results for the instances with up to 75 nodes. However, when considering larger instances with 76 nodes and more, FI is not competitive anymore. When starting from an inferior solution produced by FI the VND needs too much time and could not even be completed within the time limit of 10000 seconds. When comparing run-time we also see the advantage of using the GTSP over the FI; for most of the instances, especially for the larger ones, it pays off to invest more time to get a better starting solution so that the subsequent VND does not need so many iterations. We also applied the ML-ES to the GTSP + VND configuration and we observe a huge drop in run-time. It is clear that the resulting solution is the same as in the GTSP + VND configuration but the run-time could be reduced substantially. Only by using the ML-ES the VND is about 10 times faster on average with a peak speedup factor of 75 for instance P-n76-k4-C26-V2. During our tests when a solution is evaluated using ML-ES the procedure could be

[1] `https://www.ads.tuwien.ac.at/w/Research/Problem_Instances#Generalized_Vehicle_Routing_Problem_with_Stochastic_Demands`

terminated in the top 30% of the approximation levels where the acceleration is the largest.

Table 1. Results for the different configurations of the VND.

				FI + VND		GTSP + VND		GTSP + VND + ML-ES	
Instance	n	m	$E[nr]$	*obj*	$t[s]$	*obj*	$t[s]$	*obj*	$t[s]$
P-n19-k2-C7-V1	19	7	0,71	**112,105**	5	**112,105**	3	**112,105**	<1
P-n20-k2-C7-V1	20	7	0,68	**117,306**	4	**117,306**	<1	**117,306**	<1
P-n21-k2-C7-V1	21	7	0,64	**117,071**	3	**117,071**	<1	**117,071**	<1
P-n22-k2-C8-V1	22	8	0,73	**111,194**	10	**111,194**	5	**111,194**	<1
B-n31-k5-C11-V2	31	11	1,38	**355,729**	25	**355,729**	16	**355,729**	5
A-n32-k5-C11-V2	32	11	1,39	**386,909**	20	388,597	10	388,597	2
A-n33-k5-C11-V2	33	11	1,52	**318,028**	17	**318,028**	15	**318,028**	3
A-n33-k6-C11-V2	33	11	1,91	**367,629**	23	**367,629**	16	**367,629**	4
A-n34-k5-C12-V2	34	12	1,66	**419,124**	29	**419,124**	25	**419,124**	4
B-n34-k5-C12-V2	34	12	1,34	**363,089**	33	**363,089**	13	**363,089**	5
B-n35-k5-C12-V2	35	12	1,54	**501,470**	32	**501,470**	14	**501,470**	6
A-n36-k5-C12-V2	36	12	1,34	404,579	30	**399,905**	23	**399,905**	7
A-n37-k5-C13-V2	37	13	1,43	**359,133**	45	**359,133**	20	**359,133**	3
A-n37-k6-C13-V2	37	13	1,95	467,266	31	**430,987**	32	**430,987**	7
A-n38-k5-C13-V2	38	13	1,71	**371,795**	57	**371,795**	20	**371,795**	2
B-n38-k6-C13-V2	38	13	1,93	**386,195**	55	389,241	27	389,241	7
A-n39-k5-C13-V2	39	13	1,48	390,400	47	**371,410**	20	**371,410**	8
A-n39-k6-C13-V2	39	13	1,83	**417,844**	43	**417,844**	40	**417,844**	8
B-n39-k5-C13-V2	39	13	1,45	**281,482**	50	**281,482**	<1	**281,482**	<1
P-n40-k5-C14-V2	40	14	1,51	**214,775**	175	**214,753**	49	**214,753**	4
B-n41-k6-C14-V2	41	14	1,82	**404,261**	93	**404,261**	34	**404,261**	11
B-n43-k6-C15-V2	43	15	1,81	394,529	74	**347,650**	33	**347,650**	6
A-n44-k6-C15-V2	44	15	2,00	**505,129**	105	508,981	51	508,981	15
B-n45-k5-C15-V2	45	15	1,51	**419,613**	116	**419,613**	59	**419,613**	8
B-n45-k6-C15-V2	45	15	1,96	**358,989**	72	**358,989**	83	**358,989**	31
P-n45-k5-C15-V2	45	15	1,61	**239,568**	172	**239,357**	94	**239,357**	6
A-n45-k6-C15-V3	45	15	2,09	**478,219**	105	**478,219**	56	**478,219**	12
A-n45-k7-C15-V3	45	15	2,06	516,508	94	**488,017**	99	**488,017**	34
A-n46-k7-C16-V3	46	16	2,08	**465,624**	209	471,980	82	471,980	16
A-n48-k7-C16-V3	48	16	2,13	474,210	150	**462,548**	95	**462,548**	35
A-n53-k7-C18-V3	53	18	2,09	450,973	268	**443,875**	97	**443,875**	16
A-n54-k7-C18-V3	54	18	2,19	507,805	201	**490,544**	134	**490,544**	41
A-n55-k9-C19-V3	55	19	2,75	475,919	292	**474,048**	114	**474,048**	15
A-n60-k9-C20-V3	60	20	2,80	**614,515**	517	620,897	361	620,897	117
P-n76-k4-C26-V2	76	26	1,33	461,753	>10000	**310,397**	4312	**310,397**	58
P-n76-k5-C26-V2	76	26	1,67	373,937	>10000	**310,397**	3748	**310,397**	56
P-n101-k4-C34-V2	101	34	1,25	992,679	>10000	**371,926**	9979	**371,926**	397

Next we show how average results over 30 independent runs of the VNS with the GTSP starting solution and the ML-ES compares to an exact algorithm. The exact algorithm is the integer L-shaped method [14] applied to the GVRPSD which is a two-level approach based on a mixed integer programming model for the GTSP. Within a branch-and-cut framework it iteratively adds cuts generated in the lower level setting a lower bound on the restocking costs in the upper level. Due to space limitation this method is not described here in more detail. For the exact algorithm the optimality gap (*gap*) and the time needed is stated in the table.

Table 2. Comparison of the proposed VNS with an exact integer L-shaped method.

				L-shaped			VNS + GTSP + ML-ES		
Instance	n	m	$E[nr]$	obj	gap	$t[s]$	$\overline{obj}$	sd	$t^{*}[s]$
P-n19-k2-C7-V1	19	7	0,71	**112,105**	0,0%	<1	**112,105**	0,00	<1
P-n20-k2-C7-V1	20	7	0,68	**117,306**	0,0%	<1	**117,306**	0,00	<1
P-n21-k2-C7-V1	21	7	0,64	**117,071**	0,0%	<1	**117,071**	0,00	<1
P-n22-k2-C8-V1	22	8	0,73	**111,194**	0,0%	<1	**111,194**	0,00	<1
B-n31-k5-C11-V2	31	11	1,38	**355,729**	11,1%	>7200	**355,729**	0,00	5
A-n32-k5-C11-V2	32	11	1,39	**386,909**	0,0%	1331	**386,909**	0,00	25
A-n33-k5-C11-V2	33	11	1,52	**318,028**	0,0%	374	**318,028**	0,00	2
A-n33-k6-C11-V2	33	11	1,91	**364,589**	0,0%	598	**364,589**	0,00	27
A-n34-k5-C12-V2	34	12	1,66	**419,124**	0,0%	3719	**419,124**	0,00	4
B-n34-k5-C12-V2	34	12	1,34	**363,089**	18,1%	>7200	**363,089**	0,00	4
B-n35-k5-C12-V2	35	12	1,54	501,470	26,3%	>7200	**501,450**	0,11	6
A-n36-k5-C12-V2	36	12	1,34	**399,905**	9,4%	>7200	**399,905**	0,00	7
A-n37-k5-C13-V2	37	13	1,43	**359,133**	0,0%	98	**359,133**	0,00	3
A-n37-k6-C13-V2	37	13	1,95	434,865	18,5%	>7200	**430,987**	0,00	7
A-n38-k5-C13-V2	38	13	1,71	**371,795**	0,0%	588	**371,795**	0,00	2
B-n38-k6-C13-V2	38	13	1,93	**386,195**	12,2%	>7200	388,734	1,15	8
A-n39-k5-C13-V2	39	13	1,48	**371,410**	7,6%	>7200	**371,410**	0,00	8
A-n39-k6-C13-V2	39	13	1,83	**417,844**	5,6%	>7200	**417,844**	0,00	8
B-n39-k5-C13-V2	39	13	1,45	**281,482**	0,0%	282	**281,482**	0,00	<1
P-n40-k5-C14-V2	40	14	1,51	**214,753**	0,0%	392	**214,753**	0,00	4
B-n41-k6-C14-V2	41	14	1,82	408,977	16,7%	>7200	**404,261**	0,00	10
B-n43-k6-C15-V2	43	15	1,81	**347,650**	19,7%	>7200	**347,650**	0,00	6
A-n44-k6-C15-V2	44	15	2,00	509,254	16,2%	>7200	**508,981**	0,00	15
B-n45-k5-C15-V2	45	15	1,51	**419,613**	3,6%	>7200	419,096	1,05	8
B-n45-k6-C15-V2	45	15	1,96	367,730	23,6%	>7200	**358,989**	0,00	31
P-n45-k5-C15-V2	45	15	1,61	**239,357**	4,6%	>7200	**239,357**	0,00	6
A-n45-k6-C15-V3	45	15	2,09	478,265	16,2%	>7200	**478,219**	0,00	12
A-n45-k7-C15-V3	45	15	2,06	491,539	30,6%	>7200	**488,017**	0,00	34
A-n46-k7-C16-V3	46	16	2,08	**465,624**	19,0%	>7200	471,539	0,51	16
A-n48-k7-C16-V3	48	16	2,13	469,690	28,7%	>7200	**462,548**	0,00	35
A-n53-k7-C18-V3	53	18	2,09	**443,873**	13,6%	>7200	443,875	0,00	16
A-n54-k7-C18-V3	54	18	2,19	500,349	28,6%	>7200	**490,544**	0,00	41
A-n55-k9-C19-V3	55	19	2,75	483,997	21,7%	>7200	**474,048**	0,00	15
A-n60-k9-C20-V3	60	20	2,80	623,528	35,6%	>7200	**617,575**	4,98	118
P-n76-k4-C26-V2	76	26	1,33	**310,397**	6,1%	>7200	**310,397**	0,00	55
P-n76-k5-C26-V2	76	26	1,67	**310,397**	5,8%	>7200	**310,397**	0,00	53
P-n101-k4-C34-V2	101	34	1,25	**371,926**	5,7%	>7200	**371,926**	0,00	379

Table 2 shows the numerical comparison with the exact method. The high optimality gaps on the medium to large instances show that the GVRPSD is a hard problem but on the instances where the L-shaped method is able to find a proven optimal solution the VNS also finds it in substantially less time. In the extreme case of instance A-n34-k5-C12-V2 the VNS found an optimal solution in all 30 runs about 865 times faster than the exact algorithm. However, since the L-shaped method guarantees the optimality of the solution we cannot directly compare the run-time of these algorithms. Our tests further showed that in the VNS the ML-ES can be terminated within the top 4% of the approximation levels which is even better than for the VND.

6 Conclusions and Future Work

In this work a variable neighborhood search approach for the generalized vehicle routing problem with stochastic demands under the preventive restocking policy is presented. The problem has not yet been considered so far in the literature although many real world problems can be modelled in this way. An initial attempt to solve this hard stochastic combinatorial optimization problem was made. Therefore, concepts from both, the related VRPSD and the GTSP, are used. The solution representation and the solution evaluation method that had proved to work well for the VRPSD were adapted to the GVRPSD. On top of that a multi-level evaluation scheme was used to substantially reduce the time needed for evaluating a solution candidate. The computational results show that the VNS with the GTSP initial solution and the ML-ES is able to find optimal or near-optimal solutions in short times. Future work can include the development and improvement of the exact algorithm for the GVRPSD to get optimal solutions to more instances. In a following step also the combination of the techniques presented here, especially the ML-ES, and the methods used for solving this problem exactly should be investigated. Although the ML-ES is primarily developed for the GVRPSD applications to other similar problems like the VRPSD are also possible and could lead to a significant performance gain.

References

1. Afsar, H.M., Prins, C., Santos, A.C.: Exact and heuristic algorithms for solving the generalized vehicle routing problem with flexible fleet size. International Transactions in Operational Research 21(1), 153–175 (2014)
2. Bektaş, T., Erdoğan, G., Røpke, S.: Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. Transportation Science 45(3), 299–316 (2011)
3. Bertsimas, D.J.: Probabilistic Combinatorial Optimization Problems. Ph.D. thesis, Massachusetts Institute of Technology (1988)
4. Bianchi, L., Birattari, M., Chiarandini, M., Manfrin, M., Mastrolilli, M., Paquete, L., Rossi-Doria, O., Schiavinotto, T.: Hybrid metaheuristics for the vehicle routing problem with stochastic demands. Journal of Mathematical Modelling and Algorithms 5(1), 91–110 (2006)
5. Fischetti, M., Salazar González, J.J., Toth, P.: The symmetric generalized traveling salesman polytope. Networks 26(2), 113–123 (1995)
6. Fischetti, M., Salazar González, J.J., Toth, P.: A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Operations Research 45(3), 378–394 (1997)
7. Gendreau, M., Laporte, G., Séguin, R.: A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. Operations Research 44(3), 469–477 (1996)
8. Hà, M.H., Bostel, N., Langevin, A., Rousseau, L.M.: An exact algorithm and a metaheuristic for the generalized vehicle routing problem with flexible fleet size. Computers & Operations Research 43, 9–19 (2014)

9. Hansen, P., Mladenović, N.: Variable neighborhood search. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol. 57, pp. 145–184. Springer US (2003)
10. Hansen, P., Mladenović, N., Moreno Pérez, J.: Variable neighbourhood search: methods andapplications. Annals of Operations Research 175(1), 367–407 (2010)
11. Henry-Labordere: The record balancing problem: A dynamic programming solution of the generalized traveling salesman problem. RAIRO Operations Research B2, 43–49 (1969)
12. Hjorring, C., Holt, J.: New optimality cuts for a singlevehicle stochastic routing problem. Annals of Operations Research 86(0), 569–584 (1999)
13. Jaillet, P.: Probabilistic Traveling Salesman Problems. Ph.D. thesis, Massachusetts Institute of Technology (1985)
14. Laporte, G., Louveaux, F.V., van Hamme, L.: An Integer L-Shaped Algorithm for the Capacitated Vehicle Routing Problem with Stochastic Demands. Operations Research 50(3), 415–423 (2002)
15. Laporte, G., Louveaux, F.: Solving stochastic routing problems with the integer L-shaped method. In: Crainic, T., Laporte, G. (eds.) Fleet Management and Logistics, pp. 159–167. Springer (1998)
16. Marinakis, Y., Iordanidou, G.R., Marinaki, M.: Particle swarm optimization for the vehicle routing problem with stochastic demands. Applied Soft Computing 13(4), 1693 – 1704 (2013)
17. Marinakis, Y., Marinaki, M., Migdalas, A.: A hybrid clonal selection algorithm for the vehicle routing problem with stochastic demands. In: Pardalos, P.M., Resende, M.G., Vogiatzis, C., Walteros, J.L. (eds.) Learning and Intelligent Optimization, Lecture Notes in Computer Science, vol. 8426, pp. 258–273. Springer (2014)
18. Pop, P.C., Fuksz, L., Marc, A.: A variable neighborhood search approach for solving the generalized vehicle routing problem. In: Polycarpou, M., de Carvalho, A., Pan, J.S., Woniak, M., Quintian, H., Corchado, E. (eds.) Hybrid Artificial Intelligence Systems, Lecture Notes in Computer Science, vol. 8480, pp. 13–24. Springer (2014)
19. Pop, P.C., Kara, I., Marc, A.H.: New mathematical models of the generalized vehicle routing problem and extensions. Applied Mathematical Modelling 36(1), 97 – 107 (2012)
20. Rei, W., Gendreau, M., Soriano, P.: A hybrid monte carlo local branching algorithm for the single vehicle routing problem with stochastic demands. Transportation Science 44(1), 136–146 (2010)
21. Saskena, J.: Mathematical model of scheduling clients through welfare agencies. Journal of Canadian Operational Research Society 8, 185–200 (1970)
22. Secomandi, N., Margot, F.: Reoptimization approaches for the vehicle-routing problem with stochastic demands. Operations Research 57(1), 214–230 (2009)
23. Srivastava, S.S., Kumar, S., Carg, R.C., Sen, P.: Generalized traveling salesman problem through n sets of nodes. Canadian Operational Research Society journal 7, 97–101 (1969)
24. Yang, W.H., Mathur, K., Ballou, R.H.: Stochastic vehicle routing problem with restocking. Transportation Science 34(1), 99–112 (2000)