An Integer L-shaped Method for the Generalized Vehicle Routing Problem with Stochastic Demands

Benjamin Biesinger Bin Hu Günther Raidl

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

Abstract

In this work we consider the generalized vehicle routing problem with stochastic demands (GVRPSD). This NP-hard problem combines the clustering aspect of the generalized vehicle routing problem with the uncertainty aspect of the vehicle routing problem with stochastic demands. We propose an integer L-shaped method based on decomposition and branch-and-cut. The subproblem of computing the restocking costs is based on dynamic programming. We consider the preventive restocking strategy which is substantially harder than the standard restocking strategy used by the majority of the published articles for stochastic vehicle routing problems. Using this strategy the vehicle can make a return trip to the depot even before an actual stockout occurs and therefore save travel time. The GVRPSD has not been considered in the literature so far and this first exact solution attempt proves to be able to solve small to medium instances.

Keywords: vehicle routing problem, stochastic optimization, integer programming

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

1 Introduction

We consider the generalized vehicle routing problem with stochastic demands (GVRPSD) which is a combination of the well-known generalized vehicle routing problem (GVRP) and the vehicle routing problem with stochastic demands (VRPSD). The nodes are partitioned into clusters and the actual demand of each cluster depends on the realization of a random variable. This problem appears, e.g., in healthcare logistics when medical supplies need to be delivered to districts with several hospitals and the distributing company does not know beforehand how much supply is exactly needed.

Formally, the GVRPSD is defined on a complete undirected graph $G = (V, E)$ with node set V and edge set E. The edges $e \in E$ are weighted with distances $d_e \geq 0$. The set of nodes is partitioned into m disjoint subsets or clusters $C = \{C_0, C_1, \ldots, C_m\}, C_0, \ldots, C_m \subseteq V$, such that $C_0 \cup C_1 \cup \cdots \cup C_m = V$. Node $v_0 \in V$ is a dedicated depot and the only node of cluster C_0. Each other cluster $C_j, \forall j = 1, \ldots, m$ has an associated demand ξ_j being a random variable with expected value $E[\xi_j]$ and 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. As multiple visits are not considered we can safely assume that $p_{jk} = 0,\ \forall j = 1, \ldots, m,\ \forall k > Q$. The aim is to find a route visiting exactly one node from each cluster $C_1, \ldots, C_m$ delivering 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 depends on the actual realization of ξ_j, which becomes known only upon arrival. Possibly, the vehicle will get empty and 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 to use *a-priori* routes [6] through all clusters. For handling stockouts we adopt the *preventive* restocking strategy from the VRPSD [9,2] in which more efficient intentionally planned restocking trips back to the depot can often be performed before the actual stockout occurs, e.g., if a stockout at the next cluster is likely. However, the actual demand can still exceed the remaining vehicle capacity. Then a restock is made before satisfying the remaining demand. When using the preventive restocking strategy it is sufficient to plan one giant tour through all clusters when the problem is not further constrained as shown by Yang et al. for the VRPSD [9]. Therefore the problem is more closely related to the generalized traveling salesman problem (GTSP) rather

than to the GVRP in which more routes may be necessary.

The majority of the work for the VRPSD [4,5] only consider the standard restocking policy, in which return trips to the depot are only performed when actual stockouts occur. The preventive restocking strategy is significantly harder to consider and only metaheuristics have been applied so far [2,9]. In this work we address this problem for the generalized version and use the integer L-shaped method based on a formulation for the GTSP [3] for solving the GVRPSD exactly. This method has been introduced in [8] and is a well-known algorithm in the field of stochastic programming. It is based on a Benders decomposition approach in which subproblems compute the restocking costs [7] for which optimality cuts are then iteratively added to the initially relaxed master problem. The L-shaped method has already been successfully applied to several stochastic optimization problems, e.g,. stochastic vehicle routing problems [7,4,5].

Our model for the GVRPSD is presented in Section 2. Subproblems are solved using a dynamic programming (DP) algorithm, which is described in Section 3. The basic algorithm is improved by computing lower bounds on the restocking costs of partial solutions yielding more general optimality cuts. Computational results are shown in Section 4 and indicate that the method is effective for small to medium sized GVRPSD instances.

2 Mathematical Model

To model the GVRPSD we define binary decision variables $x_e, \forall e \in E$, which are set to one if edge e is used in the solution and zero otherwise. We further use binary variables $y_v, \forall v \in V$ to determine which nodes are visited, i.e., y_v is set to one if v is visited and zero otherwise. Finally, variable θ denotes a lower bound on the restocking costs. In a preprocessing step a global lower bound L of the expected restocking costs is computed as follows. We calculate the rounded expected number of restocks $E[nr] = \left\lfloor \frac{\sum_{k=1}^{m}(E[\xi_k])}{Q} \right\rfloor$ and for each pair of nodes (i, j) the costs $s_{ij} = d_{(i,0)} + d_{(0,j)} - d_{(i,j)}$ for a preventive restock between them. Then, L is the sum of the $E[nr]$ smallest s_{ij} values. Function $\delta(S)$ denotes the edge-set of the cut $(S, V \setminus S)$, i.e., $\delta(S) = \{(i, j) \in E \mid i \in S, j \notin S\}, \forall S \subseteq V$. Our model for the master problem works as follows. Objective function (1) minimizes the sum of the total travel costs and the additional restocking costs. Equations (2) ensure that each node which is used in the solution has exactly two outgoing edges. Constraints (3) guarantee that exactly one node from each cluster is visited. Eliminating subtours is done

by classical cut-set inequalities (4). They are dynamically added within a branch-and-cut framework. The separation procedure is based on m maximum flow computations. Inequality (5) sets the value of the lower bound of the restocking costs to be at least the global lower bound L. Inequalities (8) are the specific optimality cuts setting the restocking costs accordingly. Set Π consists of all possible cluster permutations. The edge set E_π of a cluster permutation $\pi \in \Pi$ consists of all the edges between all consecutive clusters without cluster C_0.

$$\min \quad \sum_{e \in E} d_e x_e + \theta \tag{1}$$

$$\text{s.t.} \quad \sum_{e \in \delta(\{v\})} x_e = 2y_v \qquad \text{for} \quad v \in V \tag{2}$$

$$\sum_{v \in C_j} y_v = 1 \qquad \text{for} \quad j = 1, \ldots, m \tag{3}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \forall S \subset C,\ 2 \leq |S| \leq m-2,\ C_0 \in S \tag{4}$$

$$\theta \geq L \tag{5}$$

$$x_e \in \{0,1\} \qquad \text{for} \quad e \in E \tag{6}$$

$$y_v \in \{0,1\} \qquad \text{for} \quad v \in V \tag{7}$$

$$(\theta_\pi - L) \left[\left(\sum_{(i,l) \in E_\pi} x_{il} - (m-2) \right) /2 \right] + L \leq \theta \qquad \forall \pi \in \Pi \tag{8}$$

The value of θ_π represents the recourse costs for permutation π and is computed in the subproblem via a dynamic programming procedure, which is described in the next section. This value is equal to θ for exactly the route corresponding to the cluster permutation π, because then $\sum_{(i,j) \in E_\pi} x_{ij} = m$. In all other feasible routes at least two edge variables in E_π are zero and therefore the lefthand side of the inequality does not exceed L.

3 Specific and General Optimality Cuts

In this section the computation of the restocking costs using the preventive restocking strategy is described. As mentioned before a dynamic programming procedure is used to compute these additional costs exactly. A similar algorithm has already been used for the VRPSD [9,2]. In the DP we define a recursive function $f_{ij}(q)$ for all $q = 0, \ldots, Q,\ j = 0, \ldots, m,\ i = 0, \ldots, |\pi(j)|-1$ to be the expected remaining cost of the tour after servicing the i-th node of cluster $\pi(j)$ with respect to the remaining capacity q. The auxiliary function

$b_j(l)$ returns the l-th node of cluster at position j. To compute the total cost $c^*(\pi)$ of a cluster permutation π the following DP recursion is used:

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

$$f^p_{ij}(q) = \min_{l=0,\dots,|\pi(j+1)|}\{d_{b_j(i),b_{j+1}(l)} + \sum_{k=0}^{q} f_{l,j+1}(q-k)p_{j+1,k} + \sum_{k=q+1}^{Q} 2d_{b_{j+1}(l),0} + f_{l,j+1}(q+Q-k)p_{j+1,k}\}$$

$$f^r_{ij}(q) = d_{b_j(i),0} + \min_{l=0,\dots,|\pi(j+1)|}\{d_{0,b_{j+1}(l)} + \sum_{k=0}^{Q} f_{l,j+1}(Q-k)p_{j+1,k}\}$$

$$\forall q = 0,\dots,Q,\ j = 0,\dots,m,\ i = 0,\dots,|\pi(j)|-1$$

and the boundary condition

$$f_{im}(q) = d_{b_m(i),0}, \quad \forall q = 0,\dots,Q,\ i = 0,\dots,|\pi(m)|-1$$

This DP computes for each node i and each vehicle load q if it is in the expected case more cost efficient to proceed directly to the next cluster with costs $f^p_{ij}(q)$ or to make a preventive restock with costs $f^r_{ij}(q)$. The total cost $c^*(\pi)$ is given by $f_{0,0}(Q)$. When considering restocking costs the direction of the route is important so the actual cost of an undirected route stated as permutation π is given by $c^*(\pi^*) = \min\{c^*(\pi), c^*(\text{reverse}(\pi)\}$, where reverse$(\pi)$ represents the reversed order of the elements in π. Let $\hat{c}(\pi)$ be the travel costs of permutation π from solving the model without considering the restocking costs. Then the exact restocking costs θ_π are $c^*(\pi^*) - \hat{c}(\pi)$.

Each time the master problem is solved, this DP algorithm is executed and the respective specific optimality cut (8) is added to the model if not yet included. The procedure iterates until no further cuts are separated and thus an optimum overall solution has been found. A weakness of these specific optimality cuts is that they only apply to exactly one solution but in the next section we show how to generalize these cuts in order to increase the lower bound on the restocking costs more generally.

3.1 General Optimality Cuts

We generalize the specific optimality cuts from the last section by computing lower bounds on the restocking costs for partial routes. Therefore, for a given permutation $\pi = \langle C_{\pi(0)}, C_{\pi(1)}, \dots, C_{\pi(m-1)}\rangle$ we consider $m-2$ partial segments $\pi^h = \langle C_{\pi(h)}, C_{\pi(h+1)}, \dots, C_{\pi(m-1)}\rangle$, $\forall h = 1,\dots,m-2$. Each segment implicitly starts and ends at the depot and we use a slightly modified version of the DP

for computing a lower bound on the restocking costs θ_{π^h} for all solutions using this segment. For all θ_{π^h} values which are larger than zero the following general optimality cut is added to the model.

$$(\theta_{\pi^h} - L)\left[\left(\sum_{(i,l)\in E_{\pi^h}} x_{il} - (m-h-3)\right)/2\right] + L \leq \theta$$

Adding such a general optimality cut increases the lower bounds on the restocking costs for all cluster permutations π which contain π^h as partial segment. In a naive approach $2(m-2)$ executions of the DP algorithm would be needed to compute the restocking costs of all these segments. However, by using an incremented evaluation technique we can store the values of $f_{ij}(q)$ and use them for the next DP computation. More specifically, we start with the last segment π^{m-2} and calculate $f_{i,m-2}(q),\ \forall i \in \pi(m-2),\ q = 0,\ldots,Q$. This value is used for the next iteration where segment π^{h-3} is considered and we only have to compute $f_{i,m-3}(q), \forall i \in \pi(m-3), q = 0,\ldots,Q$ and use the previous values of $f_{i,m-2}(q)$. This method saves many unnecessary computations so that we can add up to $m-2$ general optimality cuts without having to execute the whole DP algorithm each time.

4 Computational Results

The algorithm is implemented in C++ with CPLEX 12.6 and tested on a single core of an Intel Xeon with 2.53 GHz and 3GB RAM. Our benchmark instances[1] are based on the GVRP instances by Bektaş et al. [1]. We adapted them to the GVRPSD by assigning the original demand values to be the expected demands and choosing randomly 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 to be equally likely, so we assumed a uniform distribution over these values. Additionally, we used instances with a higher demand variance ($\pm 50\%$ and $\pm 80\%$, respectively).

In Table 1 the integer L-shaped algorithm with only the specific optimality cuts is compared to the version where both specific and general optimality cuts are considered. For both configurations the final objective value *obj*, the time needed in seconds $t[s]$ and the resulting optimality gap (*gap*) are listed. Additionally, the number of specific ($\#O_S$) and general ($\#O_G$) optimality cuts are given. We observe that when only the specific optimality cuts are

[1] `www.ads.tuwien.ac.at/w/Research/Problem_Instances`

Table 1
Results of the integer L-shaped method with and without general optimality cuts.

				L-shaped + OPT_S				L-shaped + OPT_S + OPT_G				
Instance	n	m	$E[nr]$	obj	$t[s]$	gap	$\#\text{O}_S$	obj	$t[s]$	gap	$\#\text{O}_S$	$\#\text{O}_G$
A-n32-k5-C11-V2	32	11	1,39	386,909	>14400	4,2%	877	386,909	2086	0,0%	160	631
A-n33-k5-C11-V2	33	11	1,52	318,028	231	0,0%	389	318,028	84	0,0%	76	353
A-n33-k6-C11-V2	33	11	1,91	364,589	10172	0,0%	1367	364,589	658	0,0%	78	359
A-n34-k5-C12-V2	34	12	1,66	419,124	>14400	8,1%	2752	419,124	10824	0,0%	260	1467
A-n36-k5-C12-V2	36	12	1,34	399,905	>14400	12,9%	2018	399,997	>14400	9,1%	671	2882
A-n37-k5-C13-V2	37	13	1,43	359,133	1271	0,0%	903	359,133	212	0,0%	158	594
A-n38-k5-C13-V2	38	13	1,71	371,795	>14400	4,0%	2007	371,795	339	0,0%	61	408
P-n40-k5-C14-V2	40	14	1,51	214,753	2308	0,0%	1279	214,753	399	0,0%	123	748
P-n45-k5-C15-V2	45	15	1,61	239,357	>14400	7,2%	2337	239,568	>14400	3,0%	281	1910
P-n76-k4-C26-V2	76	26	1,33	310,397	>14400	5,9%	1229	310,397	>14400	6,1%	283	2269
P-n76-k5-C26-V2	76	26	1,67	310,397	>14400	5,9%	1252	310,397	>14400	6,3%	299	3484
Instances with higher demand variance												
A-n32-k5-C11-V2	32	11	1,39	393,475	>14400	6,5%	1180	393,475	3845	0,0%	199	1209
A-n33-k5-C11-V2	33	11	1,52	322,048	571	0,0%	577	322,048	134	0,0%	108	605
A-n33-k6-C11-V2	33	11	1,91	366,445	9380	0,0%	1335	366,445	570	0,0%	84	519
A-n34-k5-C12-V2	34	12	1,66	427,409	>14400	9,7%	3673	427,409	>14400	2,6%	371	2705
P-n40-k5-C14-V2	40	14	1,51	215,798	2480	0,0%	1662	215,798	392	0,0%	131	1056
P-n45-k5-C15-V2	45	15	1,61	241,271	>14400	8,1%	2624	241,271	>14400	3,6%	257	2315
P-n76-k4-C26-V2	76	26	1,33	310,397	>14400	6,0%	1078	310,397	>14400	7,2%	161	1742
P-n76-k5-C26-V2	76	26	1,67	311,431	>14400	6,4%	1032	311,431	>14400	6,8%	255	3786
Instances with $E[nr] > 2$												
A-n45-k6-C15-V3	45	15	2,09	478,219	>14400	16,8%	2369	478,219	>14400	13,3%	503	4400
A-n45-k7-C15-V3	45	15	2,06	491,539	>14400	31,7%	3498	491,739	>14400	29,0%	871	6614
A-n46-k7-C16-V3	46	16	2,08	471,716	>14400	23,3%	2626	465,624	>14400	15,7%	511	4734
A-n48-k7-C16-V3	48	16	2,13	465,343	>14400	28,7%	2361	465,35	>14400	25,8%	892	8151
A-n53-k7-C18-V3	53	18	2,09	443,873	>14400	14,6%	1977	445,802	>14400	11,8%	539	6521
A-n54-k7-C18-V3	54	18	2,19	496,364	>14400	28,8%	1961	507,513	>14400	27,9%	661	6785
A-n55-k9-C19-V3	55	19	2,75	481,531	>14400	21,9%	1662	483,997	>14400	19,2%	359	4625
A-n60-k9-C20-V3	60	20	2,8	622,404	>14400	36,6%	1726	622,404	>14400	34,6%	440	5835
Average gap						10,7%				8,2%		

used only 7 out of 27 instances could be solved to optimality within the CPU time limit of four hours. This number is increased to 11 when also general optimality cuts are added. Also for the unsolved instances the final optimality gap is consistently better and the number of specific optimality cuts lower when considering general optimality cuts. We further observe that a lot of optimality cuts were needed to solve the model, which indicates a potential to improve this algorithm. Apparently, the lower bounds obtained by the general optimality cuts are still too weak to be able to solve all instances in reasonable time. We notice that the complexity of the instances does not only depend on the number of clusters and nodes but even more on the number of expected restocks as no instance with $E[nr] > 2$ could be solved to optimality.

5 Conclusions

We presented an integer L-shaped method for the GVRPSD. Neither the GVRPSD nor the VRPSD with preventive restocking have been considered by exact algorithms so far, so an initial attempt was made. Results showed that this approach is effective for solving smaller instances up to about 40 nodes and 13 clusters with $E[nr] \leq 2$. Possible directions for future research are on the one hand to increase the lower bounds of the restocking costs even further and on the other hand to increase the global lower bound to be able to cut off more solutions. Also, the number of DP executions is clearly a bottleneck of this algorithm, so reducing it could also lead to a major decrease in run-time.

References

[1] Bektaş, T., G. Erdoğan and S. Røpke, *Formulations and branch-and-cut algorithms for the generalized vehicle routing problem*, Transportation Science **45** (2011), pp. 299–316.

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

[3] Fischetti, M., J. J. Salazar González and P. Toth, *The symmetric generalized traveling salesman polytope*, Networks **26** (1995), pp. 113–123.

[4] Hjorring, C. and J. Holt, *New optimality cuts for a singlevehicle stochastic routing problem*, Annals of Operations Research **86** (1999), pp. 569–584.

[5] Jabali, O., W. Rei, M. Gendreau and G. Laporte, *Partial-route inequalities for the multi-vehicle routing problem with stochastic demands*, Discrete Applied Mathematics **177** (2014), pp. 121 – 136.

[6] Jaillet, P., *Probabilistic traveling salesman problems*, PhD thesis, MIT (1985).

[7] Laporte, G. and F. Louveaux, *Solving stochastic routing problems with the integer L-shaped method*, in: T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, Springer, 1998 pp. 159–167.

[8] Van Slyke, R. M. and W. Roger, *L-shaped linear programs with applications to optimal control and stochastic programming*, SIAM Journal on Applied Mathematics **17** (1969), pp. 638–663.

[9] Yang, W.-H., K. Mathur and R. H. Ballou, *Stochastic vehicle routing problem with restocking*, Transportation Science **34** (2000), pp. 99–112.