CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

A Local Branching Matheuristic for the Multi-Vehicle Routing Problem with Stochastic Demand

Florent Hernandez
Michel Gendreau
Ola Jabali
Walter Rei

December 2016

CIRRELT-2016-67

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

A Local Branching Matheuristic for the Multi-Vehicle Routing Problem with Stochastic Demands

Florent Hernandez[1,2,*], Michel Gendreau[1,2], Ola Jabali[3], Walter Rei[1,4]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Department of Mathematics and Industrial Engineering, Polytechnique Montréal, P.O. Box 6079, Station Centre-Ville, Montréal, Canada H3C 3A7

[3] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milano, Italy

[4] Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

Abstract. This paper proposes a local branching matheuristic for the vehicle routing problem with stochastic demands (VRPSD). The problem is cast in a two stage stochastic programming model, according to which routes are planned in the first stage and executed in the second stage. In this setting, a failure may occur if a vehicle does not have sufficient capacity to serve the realized demand of a customer, which is revealed only upon arrival at a customer's location. In the event of a failure, a recourse action is performed by having the vehicle return to the depot to replenish its capacity and resume its planned route at the point of failure. Thus, the objective of the VRPSD is to minimize the sum of the planned routes cost and of the expected recourse cost. We propose a local branching matheuristic to solve the multi-VRPSD. We introduce an intensification procedure applied at each node of the local branching tree. This procedure is embedded in a multidescent scheme for which we propose a diversification strategy. Extensive computational results demonstrate the effectiveness of our matheuristic. In particular, we found new best solutions to 63 out of 270 instances available from the literature. Furthermore, compared to a subset of instances solved to optimality by an exact algorithm, our matheuristic yielded an average gap of 0.08%.

Keywords: Local branching, stochastic vehicle routing problems, vehicle routing, stochastic demand, stochastic programming, matheuristics.

Acknowledgements. The authors gratefully acknowledge funding provided by the Natural Sciences and Engineering Research Council of Canada (NSERC).

---

* Corresponding author: florent.hernandez@cirrelt.net

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2016

1. Introduction

Given a set of geographically dispersed customers, the Vehicle Routing Problem (VRP) consists of determining vehicle routes that minimize the travel cost of the operated vehicles. The routes must adhere to the conditions that each customer is visited exactly once by a single vehicle, each vehicle starts and ends its route at a single depot, and the total demand on a route does not exceed the vehicle capacity. The VRP, and several of its variants, have been extensively studied by the operations research community (see, e.g., Toth and Vigo [29] and Golden et al. [14]). The vast majority of this literature assumes that the problem parameters are deterministic. However, in practice, problem parameters such as the presence of customers, demands, travel and service times, are often uncertain. To handle such uncertainties, several stochastic versions of the VRP have been studied (see Gendreau et al. [10] for an overview of these problems). Such stochastic problems are significantly more difficult to solve than their deterministic counterparts. Therefore, there is an overall need to develop algorithms that are able to efficiently solve large-scaled instances of these stochastic problems.

In this paper, we consider the vehicle routing problem with stochastic demands (VRPSD). In this problem, it is assumed that each customer demand follows a given probability distribution and that a specific demand is only revealed upon the arrival of a vehicle to the customer's location. Practical examples of the VRPSD are found in the delivery and collection of money to and from banks (Bertsimas [3] and Lambert et al. [19]), in home oil delivery (Chepuri and Homem de Mello [6]), beer distribution and garbage collection (Yang et al. [30]). To formulate the problem, we apply the *a priori optimization* framework (developed by Bertsimas [5], Jaillet [18] and Bertsimas et al. [4]), which produces a two-stage stochastic programming model. In the first stage, an *a priori* solution, consisting of a set of planned vehicle routes, is determined. These routes are then executed in the second stage at which time customer demands are gradually revealed. As demand values are uncertain in the first stage, a vehicle may reach a customer with insufficient residual capacity to meet the realized demand. This situation leads to a *route failure* requiring a *recourse* action. In the event of a failure, a recourse action adhering to a predetermined recourse policy is performed. Several recourse policies have been proposed for the VRPSD (see Gendreau et al. [11]). Among these, the *classical* recourse policy is one of the most widely used in the literature. Following this policy, when a failure occurs, the vehicle returns to the depot to replenish its capacity and resumes its planned route at the point of failure. The objective of the VRPSD is then to minimize the sum of both the total cost of the planned routes,

i.e., the total traveled distance, and the expected recourse cost, i.e., the expected distance traveled to perform the back and forth trips to the depot when failures occur.

A series of exact algorithms for the VRPSD with classical recourse have been proposed, see Gendreau et al. [12], Hjorring and Holt [16], Laporte et al. [21], Christiansen and Lysgaard [7], Jabali et al. [17], Gauvin et al. [9] and Leuliet [23]. A few heuristics have also been been developed for this problem. Rei et al. [26] proposed a local branching scheme combined with Monte Carlo sampling to solve the single-vehicle version of the problem. Goodson et al. [15] utilized a cyclic-order solution encoding to facilitate local search for to the VRPSD. Mendoza and Villegas [25] proposed a multi-space sampling heuristic combining a tour partitioning procedure and a set partitioning model to solve the problem. Some studies have also expanded the VRPSD with classical recourse to include other dimensions. Gendreau et al. [13] developed a tabu search heuristic for the vehicle routing problem with both stochastic customers and demands. The capacitated vehicle routing problem with stochastic demands and time windows was introduced by Lei et al. [22], who solved the problem using an adaptive large neighbourhood search heuristic. Mendoza et al. [24] studied the vehicle routing problem with stochastic demands and duration constraints, which they solved by applying a greedy randomized adaptive search procedure. Finally, it should be noted that several heuristics have also been proposed for more intricate recourse policies, such as restocking policies by Yang et al. [30], route reoptimization by Secomandi and Margot [27] and pairing strategies by Ak and Erera [1].

The purpose of this paper is to develop an efficient local branching matheuristic for the multi vehicle VRPSD defined using the classical recourse policy. The choice of this specific policy is motivated by the fact that it is: 1) easily applied in practice, requiring no particular information and communication technologies to be used to coordinate the fleet of vehicles, and 2) can be readily used to define more complex policies, e.g., restocking policies by Yang et al. [30]. Furthermore, the classical recourse policy defines a benchmark on which any other policy can be compared to assess its value. As for the specific methodology used to design the proposed matheuristic we use local branching, which was originally developed by Fischetti and Lodi [8] and essentially works on solving a combinatorial optimization problem by exploring, using an exact solver, reduced portions of the solution space. A branching scheme that is based on the hamming distance was proposed by the authors to perform the overall search. Therefore, local branching enables exact solution methods to be more efficiently applied by performing a partial search of the feasible region of the considered model and thus heuristically solve larger instances. By guiding the search process towards more promising regions, this type of method can rapidly produce high quality solutions. In

the case of the VRPSD, considering the various exact methods that have been developed, local branching clearly appeared as an appropriate strategy to design a matheuristic for the VRPSD with the classical recourse policy. The algorithms of Jabali et al. [17] and Leuliet [23] are specifically used in the method that we propose. Furthermore, it was previously shown by Rei et al. [26], that local branching produced excellent results in the case of the single vehicle VRPSD. In the present paper, we look to generalize this approach for the case where a fleet of m vehicles (i.e., $m > 1$) are available to solve the VRPSD.

The present paper makes three scientific contributions. We first propose a local branching matheuristic for the multi vehicle VRPSD, that is based on a multi-descent search process and that applies a strategy to dynamically allocate the available computing resources. Secondly, we develop a new intensification procedure that is applied at each node of the local branching search tree. Finally, we develop a specialized diversification strategy to instantiate the different descents carried out and that greatly expands the overall search process that is performed by our matheuristic. The added value of these contributions is validated through an extensive computational analysis performed on a diversified set of instances (i.e., instances of varying size and complexity). The numerical results obtained highlight the effectiveness of our matheuristic in terms of both solution quality and computational speed. It should also be noted that while the developed matheuristic is used to solve the VRPSD, it can be directly applied to any stochastic variant of the VRP for which an exact solution method is available.

The remainder of this paper is organized as follows. In Section 2 we present the formulations of the VRPSD formulation, and briefly discuss the integer L-shaped algorithm used for solving it. In Section 3 we introduce our local branching matheuristic followed by computational results in Section 4. Finally, in Section 5 we present our conclusions.

2. The vehicle routing problem with stochastic demand

In Section 2.1 we present the two-stage stochastic programming formulation of the VRPSD, which was initially proposed by Laporte et al. [21]. We then briefly describe the integer L-shaped algorithm in Section 2.2, which is used to solve the subproblems generated through the multi-descent search process described in Sections 3.

2.1 The VRPSD Model

We formulate the VRPSD under the classical recourse policy using as a two-stage stochastic programming model. The VRPSD is defined on a complete undirected graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is the vertex set and $E = \{(v_i, v_j) : v_i, v_j \in V,\ i < j\}$ is the edge set. The depot is represented by vertex v_1 and the customers are represented by $\{v_2, \ldots, v_n\}$. Each edge $(v_i, v_j) \in E$ has a travel cost of c_{ij}. There are m vehicles at the depot, each of which has a capacity of D. We assume that the customer demands are identically and independently distributed. Each customer has a stochastic demand ξ_i with a mean of μ_i. The vehicle routes are planned in the first stage, these must start and end at the depot and each customer must be visited once by a single vehicle. Furthermore, similar to Laporte et al. [21], we assume that the total expected demand of each route should not exceed the vehicle capacity. The recourse cost associated with the routes determined in the first stage is evaluated in the second stage. Thus, the objective function is the sum of the cost of the planned routes of the first-stage and the expected recourse cost of the second-stage. The integer decision variables x_{ij} $(i < j)$ are equal to the number of times edge (v_i, v_j) appears in the first-stage solution, where x_{ij} should be interpreted as x_{ji} whenever $i > j$. If $i > 1$, then x_{ij} can only take the values 0 or 1; if $i = 1$, then x_{ij} may take the values 0, 1 or 2, where $x_{1j} = 2$ represents a route that includes a single customer v_j. The VRPSD model is then

$$\text{Minimize} \sum_{i<j} c_{ij} x_{ij} + \mathcal{Q}(x) \tag{1}$$

subject to

$$\sum_{j=2}^{n} x_{1j} = 2m, \tag{2}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2, \qquad (k = 2, \ldots, n), \tag{3}$$

$$\sum_{v_i, v_j \in S} x_{ij} \leq |S| - \left\lceil \sum_{v_i \in S} \mu_i / D \right\rceil, \qquad (S \subset V \setminus \{v_1\}, 2 \leq |S| \leq n-2) \tag{4}$$

$$0 \leq x_{ij} \leq 1 \qquad (2 \leq i < j \leq n), \tag{5}$$

$$0 \leq x_{1j} \leq 2 \qquad (j = 2, \ldots, n), \tag{6}$$

$$x = (x_{ij}) \text{ integer} \qquad (1 \leq i < j \leq n). \tag{7}$$

Constraints (2) and (3) are the degree constraints imposed for each vertex. Constraints (4) eliminate subtours and guarantee that the expected demand of each route respects the

vehicle capacity. Finally, the necessary limits on the values of the variables are imposed in constraints (5) and (6), while constraints (7) impose the integrality requirements. As for the computation of $\mathcal{Q}(x)$, it is separable by route. Furthermore, the recourse cost of a given a route r depends on its orientation $\delta = 1, 2$. We thus compute the cost of a route for each of the two orientations and select the lowest one. Let $\mathcal{Q}^{r,\delta}$ be the expected recourse cost of a given route r, where $r = 1, \ldots, m$, defined by $(v_{i_1} = v_1, v_{i_2}, \ldots, v_{i_{t+1}} = v_1)$ for orientation δ. The computation of $\mathcal{Q}(x)$ follows,

$$\mathcal{Q}(x) \quad = \quad \sum_{r=1}^{m} \min\{\mathcal{Q}^{r,1}, \mathcal{Q}^{r,2}\}, \tag{8}$$

As in Laporte et al. [21], the computation of $\mathcal{Q}^{r,1}$ is

$$\mathcal{Q}^{r,1} \quad = \quad 2\sum_{j=2}^{t}\sum_{l=1}^{j-1} P\Big(\sum_{s=1}^{j-1}\xi_{i_s} \leq lD < \sum_{s=1}^{j}\xi_{i_s}\Big)c_{1i_j}. \tag{9}$$

The above computation sums the expected recourse for each customer v_{i_j} that is scheduled on route r. This is done by sequentially considering the probabilities of incurring the l^{th} failure (where $l = 1, \ldots, j-1$) at customer v_{i_j}. The value of $\mathcal{Q}^{r,2}$ is computed similarly, by simply reversing the orientation of the route.

2.2 The integer L-shaped algorithm

As previously stated, the local branching search strategy is based on the principle of reducing the solution space around given solutions to the considered problem and then solving the resulting subproblems using an exact method. In the matheuristic developed in the present paper, the local branching subproblems are solved using the integer L-shaped method, which was put forward by Laporte and Louveaux [20]. This algorithm extends the L-shaped method of Van Slyke and Wets [28] for continuous stochastic programs which, in turn, is an application of Benders decomposition [2] to stochastic programming.

The integer L-shaped algorithm solves the VRPSD by first relaxing constraints (4) and (7). The recourse cost $\mathcal{Q}(x)$ is replaced with a variable Θ, which is initially set greater than or equal to a general lower bound on the expected cost of recourse L. Using intermediate infeasible solutions lower bounding functionals on Θ are generated. The actual recourse

cost $\mathcal{Q}(x)$ is evaluated only upon reaching promising solutions. We use the lower bounding functionals proposed by Leuliet [23]. The subtour elimination constraints are generated when they are found to be violated, and integrality is ensured by branching. Optimality cuts are generated at feasible integer solutions. For a detailed description of the integer L-shaped algorithm for the VRPSD, the reader is referred to Jabali et al. [17].

3. Local branching for the VRPSD

In this section we describe the proposed local branching matheuristic for the VRPSD. The matheuristic follows a multi-descent scheme, where during each descent a local branching search is performed. The starting point of each descent is established by a solution obtained through solving a deterministic VRP model. We first describe the general descent structure for which we propose an intensification scheme. We then introduce the multi-descent strategy followed by an intensification procedure for the initialization of the descents. Finally, we describe our scheme for the dynamic management of computational times.

3.1 General descent structure

We recall the general local branching scheme introduced by Fischetti and Lodi [8]. In principle, local branching partitions the solution space of a given MIP in a manner that yields sub problems which are relatively easy to solve. To simplify the notation used for the presentation of the algorithm, we redefine the VRPSD model as follows:

$$\begin{aligned} P(x) \quad &= \quad min\ c^{\top}x + \mathcal{Q}(x) \\ \text{subject to} \quad & \\ & x \in X, \end{aligned} \tag{10}$$

where x is a $|V| \times |V|$ matrix, and X is the set of feasible solutions to the VRPSD. The local branching scheme iteratively separates the solution space with respect to a reference solution $\bar{x}$. Let

$$\Delta(x, \bar{x}) = \sum_{(v_i,v_j)\in\bar{T}} (1 - x_{ij}) + \sum_{(v_i,v_j)\in E\setminus\bar{T}} x_{ij} \tag{11}$$

be the Hamming distance between x and $\bar{x}$ where $\bar{T} = \{(v_i, v_j) \in E | \bar{x}_{ij} = 1\}$. Given a

solution $\bar{x}$, local branching separates the solution space into two subspaces. One subspace where the distance $\Delta(x, \bar{x})$ between each solution in this subspace and $\bar{x}$ is less or equal to a given parameter κ, the complement of this subspace defines the other subspace. When κ is fixed to a small value, the solution space of the first subspace becomes rather small. Thus the corresponding subproblem can be efficiently solved. Casting this principle within a branching scheme entails progressively locally branching based on various reference solutions, corresponding to levels.

Let $\bar{x}^{v-1}$ be the optimal solution found at the left branch of level $v-1$. The left branch of level v is defined by constraining the solution space with $\Delta(x, \bar{x}^i) \geq \kappa+1 \; \forall \bar{x}^i \in \{\bar{x}^0, \cdots, \bar{x}^{v-2}\}$ and $\Delta(x, \bar{x}^v) \leq \kappa$. The right branch at level v, imposes $\Delta(x, \bar{x}^i) \geq \kappa+1 \; \forall \bar{x}^i \in \{\bar{x}^0, \cdots, \bar{x}^{v-1}\}$ upon the original problem. According to the procedure initially proposed by Fischetti and Lodi [8], the left branches are successively solved as long as the solution is improved, otherwise the right branch is solved. Formally, the left branch of level v is defined as follows:

$$
\begin{aligned}
P_v(x) \quad = \quad & min \; c^\top x + \mathcal{Q}(x) && (12)\\
s.t. \quad & x \in X && (13)\\
& \Delta(x, \bar{x}^i) \geq \kappa + 1, \quad \forall \bar{x}^i \in \{\bar{x}^0, \cdots, \bar{x}^{v-2}\}, && (14)\\
& \Delta(x, \bar{x}^{v-1}) \leq \kappa, && (15)
\end{aligned}
$$

$P_v(x)$ is referred to as the current local branching sub-problem. Constraints (14) include all right branch derived from solutions $\bar{x}^0, \cdots, \bar{x}^{v-2}$, where $\bar{x}^0$ is the initial reference solution and $\bar{x}^i$ with $i \in \{1, \cdots, v-2\}$ is the reference solution found by solving the sub-problem $P_i(x)$. Constraints (15) limit the search space to the neighbourhood of $\bar{x}^{v-1}$, which is the solution found at the previous level. A representation of the local branching tree is provided in Figure 1.

The initial local branching procedure has no time limit on solving the sub-problems. This may be impractical in stochastic problems, which may require long run times for solving restricted cases. Furthermore, according to this procedure a non improving solution found in the left branch will trigger the solution of the corresponding right branch problem. In practice this can be very time consuming in stochastic problems. To counter these issues, we propose a time limit t_{max} for the solution of each sub-problem (the computation of which is explained in Section 3.5) and we allow non improving solutions. Moreover, we allow for a number of local branching descents each of which solves u sub-problems. In what follows we

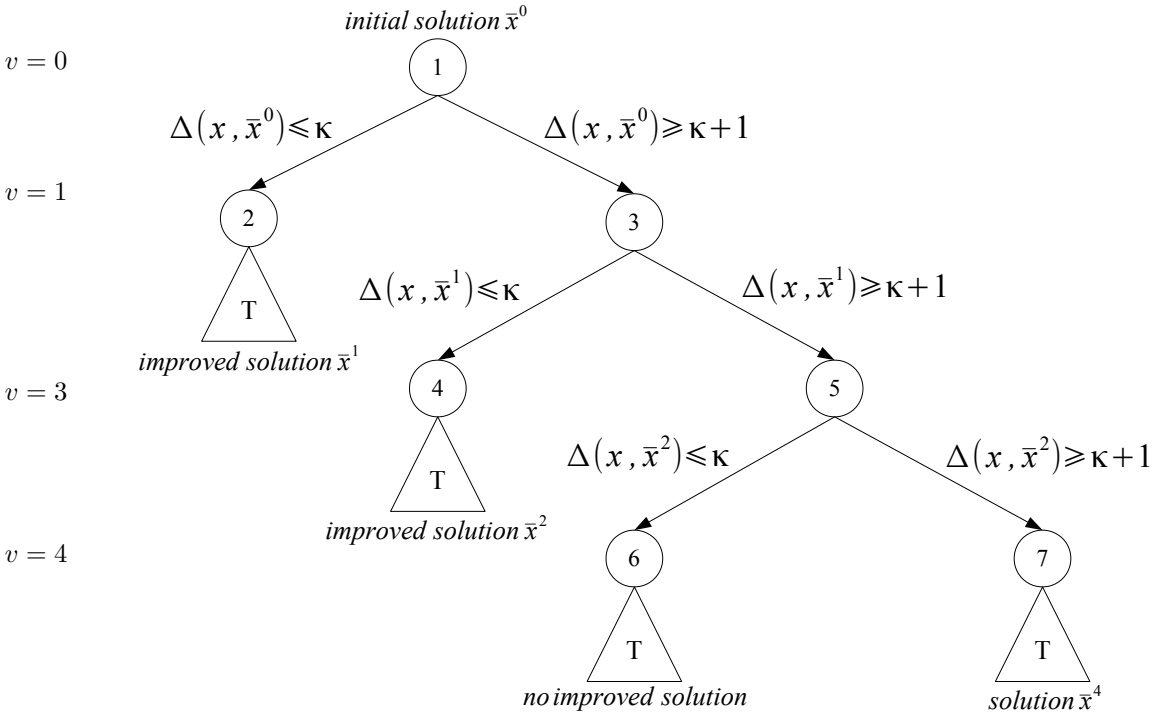

Figure 1: General local branching principle

present four potential outcomes of solving a left branch sub-problem, and their corresponding actions.

1. optimal: then the right branch is partitioned and the next left branch is solved
2. feasible: then the right branch is partitioned and the next left branch is solved
3. infeasible: then the descent is terminated, and another is initiated
4. no solution obtained within t_{max}: then the left branch is resolved with $\frac{\kappa}{2}$

Our main methodological contributions are presented in the following sections.

3.2 Intensification procedure

It was observed by Rei et al. [26] that a substantial portion of the objective function value of the VRPSD is attributed to the first stage cost, i.e., $c^\top x$. Thus, a solution with a high travel cost is unlikely to be optimal for the VRPSD. Therefore, we propose an intensification procedure based on the solutions to the VRP. We note that omitting the recourse cost from the objective function of the VRPSD yields a deterministic VRP, where the expected value

of a customer's demand is considered as its deterministic demand. For each left branch sub-problem v we define,

$$\begin{aligned} \hat{P}_v(x) \quad &= \quad min\ c^\top x && (16)\\ s.t. \quad & x \in X \\ & \Delta(x, \bar{x}^i) \geq \kappa + 1, \quad \forall \bar{x}^i \in \{\bar{x}^0, \cdots, \bar{x}^{v-2}\}, && (17)\\ & \Delta(x, \bar{x}^{v-1}) \leq \kappa. && (18) \end{aligned}$$

Let $\hat{x}^v$ be the optimal solution of $\hat{P}_v(x)$. The intensification procedure works with the following modified version of $P_v(x)$.

$$\begin{aligned} P_v(x) \quad &= \quad min\ c^\top x + \mathcal{Q}(x) \\ s.t. \quad & x \in X \\ & \Delta(x, \bar{x}^i) \geq \kappa + 1, \quad \forall \bar{x}^i \in \{\bar{x}^0, \cdots, \bar{x}^{v-2}\}, \\ & \Delta(x, \bar{x}^{v-1}) \leq \kappa, \\ & \Delta(x, \hat{x}^v) \leq \kappa && (19) \end{aligned}$$

The intensification is done by adding constraint (19), which forces solutions to be in the neighbourhood of $\hat{x}^v$. A graphical representation of the intensification procedure is illustrated in Figure 2. The intensification procedure limits the solution spaces of each left branch. While this limitation is likely to accelerate the solution process of the left branches, it may discard promising solutions. Therefore, constraint (19) is only added locally at v and is removed in subsequent sub-problems.

3.3 Multi-descent Scheme

We adopt a multi descent structure according to which each descent solves at most u sub-problems. A similar scheme was proposed by Rei et al. [26] for the single vehicle VRPSD. Let $\bar{x}^{k^1} \cdots \bar{x}^{k^u}$ be the reference solutions found during descent k and $\bar{x}^{k^0}$ be the initial solution of descent k. Where $\bar{x}^{k^0}$ is the solution to the deterministic VRP of the unexplored

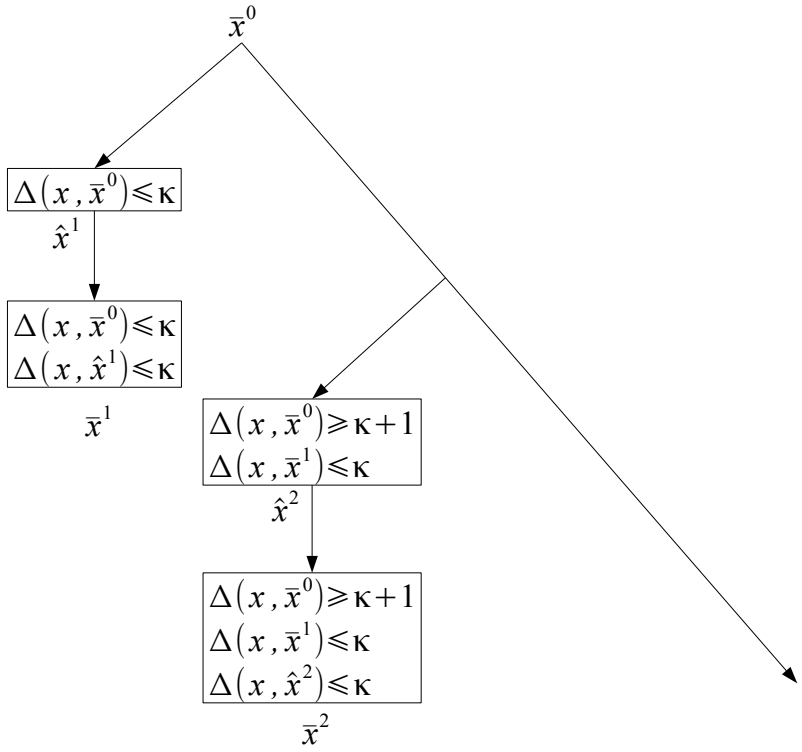

Figure 2: Local branching descent with intensification procedure

solution space. Therefore, $\bar{x}^{l^0}$ is obtained by solving the following problem.

$$\hat{P}_0^l(x) \quad = \quad min\ c^\top x \tag{20}$$

$$s.t. \quad x \in X \tag{21}$$

$$\Delta(x, \bar{x}^{k^i}) \geq \kappa + 1, \quad \forall k \in \{0, ..., l-1\}, \quad \forall i \in \{0, ..., u\} \tag{22}$$

Constraints (22) define the unexplored solution space by the end of descent $l-1$. In order to avoid revisiting solutions in a multi-descent scheme, the left branch problem of descent l $P_v^l(x)$ is defined as follows.

$$
\begin{aligned}
P_v^l(x) \quad &= \quad min\ c^\top x + \mathcal{Q}(x) \\
&s.t. \quad x \in X \\
&\Delta(x, \bar{x}^{l^i}) \geq \kappa + 1, \quad \forall \bar{x}^{l^i} \in \{\bar{x}^{l^0}, \cdots, \bar{x}^{l^{v-2}}\}, \\
&\Delta(x, \bar{x}^{l^{v-1}}) \leq \kappa, \\
&\Delta(x, \hat{x}^{l^v}) \leq \kappa \\
&\Delta(x, \bar{x}^{k^i}) \geq 1, \quad \forall k \in \{0, ..., l-1\}, \quad \forall i \in \{0, ..., u\} \qquad (23) \\
&\Delta(x, \hat{x}^{k^i}) \geq 1, \quad \forall k \in \{0, ..., l-1\}, \quad \forall i \in \{0, ..., u\} \qquad (24)
\end{aligned}
$$

In each descent, constraints (23) and (24) ensure not visiting previously found solutions to the stochastic subprolems and the deterministic subproblems, respectively. An example of the multi-descent structure is illustrated in Figure 3.

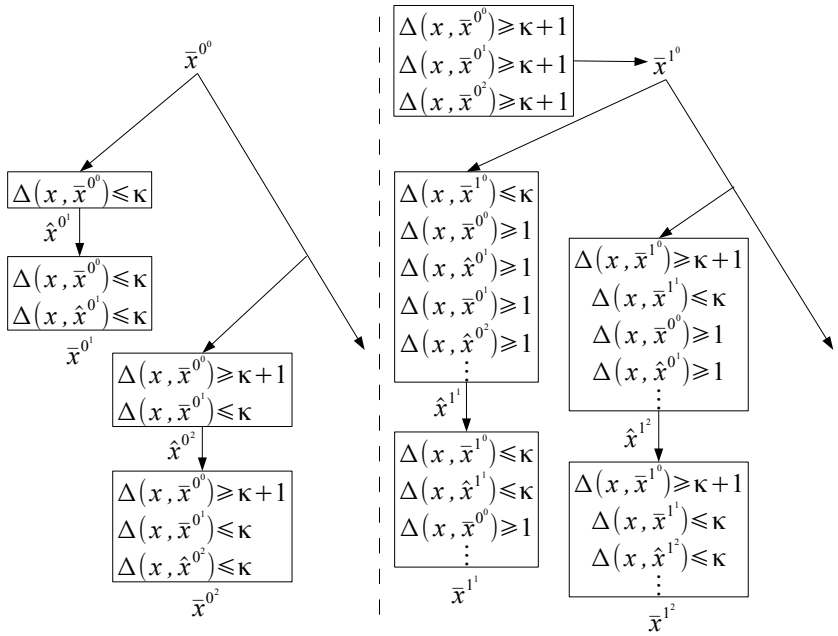

Figure 3: Multi-descent local branching with intensification

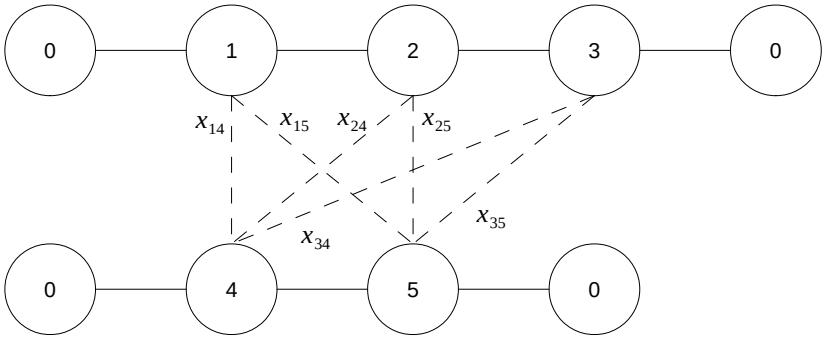

Figure 4: Example of the diversification constraints

3.4 Diversification procedure

In order to explore different regions of the solution space, we introduce a diversification procedure applied in the computation of the initial solution of each descent l, i.e., $\hat{P}_0^l(x)$. The diversification imposes transferring a client from one route to another. Let φ_r^x be the set of customers delivered by route r in solution x. The diversification procedure is achieved by solving the following.

$$\begin{aligned} \hat{P}_0^l(x) \quad &= \quad min\ c^\top x \\ &s.t. \quad x \in X \\ &\qquad \Delta(x, \bar{x}^{k^i}) \geq \kappa + 1, \quad \forall k \in \{0, ..., l-1\}, \quad \forall i \in \{0, ..., u\} \\ &\qquad \sum_{v_p, v_q \in S(\bar{x}^{k^i})} x_{pq} \geq 1 \quad \forall k \in \{0, ..., l-1\}, \quad \forall i \in \{0, ..., u\} \end{aligned} \tag{25}$$

where $\bar{x}^{k^i}$ is the solution to left branch i of descent k, and $S(x) = \{(v_p, v_q) \in E | v_p \in \varphi_r^x, v_q \in \varphi_{r'}^x, r \neq r'\}$. Figure 4, illustrates an example of constraint (25), which yields $x_{14} + x_{15} + x_{24} + x_{25} + x_{34} + x_{35} \geq 1$.

3.5 Dynamic allocation of computation times

As is common in many matheuristics, we set a computational time limit of T_{max} on the entire algorithm. Considering the proposed multi decent structure, we devise an adaptive computational time management procedure. Given a predetermined number of descents s, and an allowable number of levels u for each descent, we initially determine a time limit for each node as $t_{max} = T_{max}/(su)$. After concluding a decent, t_{max} is updated by dividing the remaining time with remaining number of potential nodes.

If the matheuristic performes all s descents and the remaining time is greater than zero, then the remaining time is used to perform another decent, i.e., the remaining time is divided amongst u subproblems of decent $s+1$. This process is repeated as long as the remaining time is greater than zero.

4. Computational results

Our experiments were performed using instances introduced in Jabali et al. [17]. The instances were generated based on the same principles proposed in Laporte et al. [21]. Namely, n vertices were generated in $[0, 100]^2$, following a continuous uniform distribution. Also, five rectangular obstacles in $[20, 80]^2$ were generated, each having a base of 4 and a height of 25, covering 5% of the entire area. The demand of each customer i follows a normal distribution $\mathcal{N}(\mu_i, \sigma_i)$ truncated at zero. All demands are independently distributed with a coefficient of variation of 30%. Let $\bar{f} = \sum_{i=1}^{n} \frac{\mu_i}{mD}$ be the average vehicle loading. Table 1 shows the considered parameter combinations. Ten instances were generated for each combination, yielding a total of 270 instances.

m	n	$\bar{f}$
2	60, 70, 80	90%, 92.5%, 95%
3	50, 60, 70	85%, 87.5%, 90%
4	40, 50, 60	80%, 82.5%, 85%

Table 1: Parameter combinations used in the instances

The algorithms described in this paper were coded in C++ and solved using IBM ILOG CPLEX 12.5. All experiments are performed on an Intel(R) Xeon(R) CPU X5675 with 12-Core 3.07 GHz and 96 GB of RAM by using a single thread.

The local branching matheuristics described in the previous sections require determining a number the following parameters: the initial number of descents, the maximum number of subproblems solved at each descent, the Hamming distance threshold κ as well as the runtime limit. Our primary aim is to examine the added value of the proposed intensification and intensification procedures. To this end we have fixed the initial number of descents to two, and the maximum number of subproblems solved at each descent to three. We experimented with κ values of 8 an 10, along with runtime limits of 900, 1,800 and 3,600 seconds. We apply the L-shaped described in Leuliet [23] to solve the stochastic programming model of each subproblem. In order to benchmark the performance of the discussed matheuristics, we ran the exact algorithm for a maximum of 10,000 seconds. Once the time limit was reached, the best

integer solution was retrieved. In order to asses the benefits of the proposed intensification procedure and diversification procedure, we ran two sets of experiments with each procedure in separation. Finally, we ran an experimental set combining both procedures.

In section 4.1 we examine the impact of the intensification and diversification procedures. In section 4.2 we present a general performance comparison between the proposed matheuristics and the exact algorithm.

4.1 Impact of intensification and diversification procedures

We summarize the results for $\kappa = 8$ on all 270 instances in Table 2. In Table 3 we present the results only on the 98 instances solved to optimality by the exact algorithm. In these tables, each row summarizes the results a class of instances, the first column describes the instances, for example "40 4 0.8" relates to instances with 40 nodes, 4 vehicles and a fill rate of 80%. The second column contains the number of instances solved to optimality, while the third column depicts the average total cost of the considered solutions by the exact algorithm. The subsequent three groups of four columns indicate the average gaps obtained when setting the run time limit to 900, 1,800 and 3,600 seconds, respectively. Each group contains four columns, where "B" corresponds to the basic multi-descent implementation (without intensification nor diversification), "I"corresponds to the multi-descent implementation with intensification, "D" corresponds to the multi-descent implementation with diversification and "DI" corresponds to the multi-descent implementation with diversification and intensification.

In Table 2 the gap between the best solution of the matheuristic is compared to the best obtained solution by the exact algorithm. Therefore, a negative entry indicates that the average performance of a matheuristic algorithm is better than that of the exact algorithm. Over the three runtime options, on average the basic implementation is substantially outperformed by "DI". The intensification procedure accelerates the solution of the sub problems by further limiting the solution space of the subproblem. However, this may compromise the solution quality. Comparing the "B" with "I", we note that the latter substantially improves the results for runtimes of 1,800 and 3,600 seconds, whereas the performance of "I" is rather comparable to that of "B" for the lower runtime of 900 seconds. Moreover, on average the performance of "D" is slightly superior to that of "I" for all three runtime options.

In Table 3 the performance of the matheuristic algorithms is compared only with instances solved to optimality. We observe that the combination of both diversification and intensification, i.e., "DI", substantially outperforms the other matheuristics for a runtime

of 3,600 seconds, yielding an average gap of 0.08% from the 98 instances solved to optimality. Considering a maximum runtime of 1,800 seconds, the performance of "DI" is better than that of the other matheuristics. Considering a runtime of 900 seconds, the average performance of "DI" is similar to that of "B".

In both tables 2 and 3, we observe that the combination of the diversification and intensification procedures yields substantially better performance compared to when considering each procedure in isolation. From Table 3 we observe that the matheuristic "DI" with 3,600 seconds as well as with 1,800 seconds found the optimal solutions on all instances with two vehicles, aside from "80 2 0.90".

We summarize the results for $\kappa = 10$ on all 270 instances in Table 4. In Table 5 we present the results only on the 98 instances solved to optimality by the exact algorithm. We observe similar patterns as in the case of $\kappa = 8$. Furthermore, considering all instances with runtimes of 900 and 1,800 seconds, matheuristic "DI" with $\kappa = 8$ slightly outperforms matheuristic "DI" with $\kappa = 10$. The performance of both matheuristics is equivalent when considering a runtime of 3,600 seconds. When considering instances solved to optimality, "DI" with $\kappa = 8$ outperforms that of $\kappa = 10$ for runtimes of 900 and 1,800 seconds.

	Exact for T_{max}=10,000s		Gap (%) for T_{max}=900s				Gap (%) for T_{max}=1,800s				Gap (%) for T_{max}=3,600s			
Instance	#OPT	Cost	B	I	D	DI	B	I	D	DI	B	I	D	DI
40 4 0.80	4	684.69	0.27	0.35	0.45	0.14	0.50	0.18	0.24	0.24	0.30	0.11	0.26	0.16
40 4 0.82	5	688.40	0.46	0.50	0.07	-0.05	-0.05	0.43	0.16	0.12	-0.03	0.05	0.00	0.30
40 4 0.85	1	699.43	-0.13	-0.22	0.43	-0.48	0.05	0.01	0.23	-0.33	0.01	-0.24	-0.10	-0.56
50 3 0.85	5	677.33	0.20	0.31	0.16	0.35	0.28	0.19	0.17	0.10	0.14	0.10	0.13	0.07
50 3 0.88	4	685.36	0.46	0.81	0.44	0.15	0.74	0.76	0.48	0.32	0.70	0.74	0.51	0.30
50 3 0.90	1	671.28	0.64	0.67	0.56	0.53	0.64	0.52	0.60	0.56	0.67	0.35	0.55	0.43
50 4 0.80	5	741.10	1.00	1.25	1.00	0.72	0.93	0.82	0.72	0.71	0.71	0.71	0.64	0.37
50 4 0.82	1	751.49	0.18	0.25	0.06	0.25	0.09	0.27	0.13	0.23	0.01	0.26	0.23	0.17
50 4 0.85	1	745.02	1.74	0.63	1.21	1.03	1.93	0.60	0.92	0.81	0.47	-0.02	0.41	-0.01
60 2 0.90	10	675.81	0.00	0.02	0.00	0.02	0.00	0.02	0.00	0.00	0.00	0.02	0.00	0.00
60 2 0.93	9	662.67	0.00	0.03	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.03	0.00	0.00
60 2 0.95	4	677.86	0.02	0.01	0.00	0.01	0.11	0.01	0.00	0.00	0.00	0.01	0.00	0.00
60 3 0.85	6	714.89	0.22	0.04	0.19	0.12	0.28	0.02	0.06	0.09	0.17	-0.01	-0.01	0.00
60 3 0.88	2	738.69	0.05	0.03	0.04	0.04	0.02	0.00	-0.01	0.01	0.05	0.00	0.00	0.03
60 3 0.90	1	723.39	0.07	0.14	0.17	0.04	0.05	0.04	0.00	-0.07	0.04	-0.06	-0.03	-0.07
60 4 0.80	0	769.84	0.88	1.16	1.38	1.11	1.33	1.14	1.36	1.15	0.95	1.12	1.13	1.13
60 4 0.82	3	766.24	0.45	0.70	0.28	0.27	0.66	0.39	0.15	0.28	0.50	0.71	0.50	0.06
60 4 0.85	0	784.43	0.46	0.34	0.43	0.42	0.36	0.30	0.39	0.21	0.17	0.33	0.32	0.13
70 2 0.90	9	708.11	0.00	0.03	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.03	0.00	0.00
70 2 0.93	6	716.16	0.26	0.22	0.34	0.03	0.21	0.24	0.04	0.05	0.21	0.21	0.05	0.05
70 2 0.95	1	718.33	0.11	0.10	0.00	0.00	0.10	0.10	0.00	0.00	0.10	0.07	0.00	0.00
70 3 0.85	7	743.42	0.15	0.17	-0.09	-0.02	0.10	-0.01	0.13	-0.12	0.13	0.08	-0.13	-0.12
70 3 0.88	2	773.06	0.25	0.33	0.23	0.27	0.26	0.23	0.21	0.28	0.29	0.33	0.15	0.26
70 3 0.90	0	777.11	0.58	0.51	0.47	0.51	0.57	0.31	0.48	0.47	0.39	0.46	0.28	0.33
80 2 0.90	8	749.15	0.08	0.07	0.07	0.06	0.07	0.07	0.07	0.01	0.06	0.06	0.04	0.03
80 2 0.93	3	773.53	0.07	0.08	0.15	0.07	0.08	0.07	0.04	0.07	0.07	0.07	0.04	0.03
80 2 0.95	0	774.35	0.17	0.12	0.16	0.01	0.19	0.18	0.08	0.05	0.18	0.16	0.09	-0.02
AVG GAP			0.32	0.32	0.31	0.21	0.36	0.26	0.25	0.20	0.24	0.22	0.19	0.12

Table 2: Gaps (%) on all instances with $\kappa = 8$

	Exact for T_{max}=10,000s		Gap (%) for T_{max}=900s				Gap (%) for T_{max}=1,800s				Gap (%) for T_{max}=3,600s			
INSTANCE	#OPT	Cost	B	I	D	DI	B	I	D	DI	B	I	D	DI
40 4 0.80	4	666.56	0.00	0.05	0.05	0.05	0.00	0.05	0.00	0.05	0.00	0.05	0.00	0.05
40 4 0.82	5	678.72	0.03	0.27	0.03	0.00	0.03	0.00	0.24	0.00	0.03	0.00	0.03	0.00
40 4 0.85	1	640.15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50 3 0.85	5	689.29	0.12	0.40	0.10	0.48	0.22	0.27	0.05	0.09	0.05	0.09	0.05	0.03
50 3 0.88	4	702.44	0.31	0.27	0.31	0.10	0.31	0.27	0.31	0.04	0.27	0.27	0.27	0.04
50 3 0.90	1	662.49	0.09	0.00	0.09	0.00	0.09	0.00	0.09	0.00	0.00	0.00	0.09	0.00
50 4 0.80	5	747.93	1.29	1.46	1.20	1.17	1.25	1.16	0.92	1.17	1.08	1.16	0.99	0.53
50 4 0.82	1	739.01	1.07	0.92	1.57	0.92	1.07	1.07	1.07	1.07	0.80	1.07	0.80	1.07
50 4 0.85	1	718.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60 2 0.90	10	675.81	0.00	0.02	0.00	0.02	0.00	0.02	0.00	0.00	0.00	0.02	0.00	0.00
60 2 0.93	9	662.29	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60 2 0.95	4	680.51	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60 3 0.85	6	721.10	0.30	0.10	0.29	0.22	0.15	0.04	0.00	0.09	0.16	0.01	0.00	0.03
60 3 0.88	2	733.91	0.00	0.00	0.07	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.07	0.00
60 3 0.90	1	702.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60 4 0.82	3	767.43	0.26	0.51	0.43	0.58	0.29	0.03	0.16	0.51	0.30	0.51	0.03	0.51
70 2 0.90	9	709.10	0.00	0.04	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.03	0.00	0.00
70 2 0.93	6	717.80	0.13	0.20	0.13	0.00	0.14	0.20	0.02	0.00	0.12	0.11	0.00	0.00
70 2 0.95	1	707.52	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
70 3 0.85	7	736.38	0.21	0.25	0.18	0.24	0.18	0.21	0.22	0.10	0.22	0.22	0.13	0.10
70 3 0.88	2	762.07	0.00	0.10	0.00	0.00	0.00	0.02	0.00	0.02	0.00	0.02	0.00	0.02
80 2 0.90	8	756.68	0.10	0.10	0.09	0.09	0.09	0.09	0.09	0.02	0.09	0.09	0.06	0.05
80 2 0.93	3	772.41	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AVG GAP			0.16	0.20	0.17	0.16	0.15	0.14	0.12	0.11	0.13	0.14	0.10	0.08

Table 3: Gaps (%) on instances solved to optimality with $\kappa = 8$

	Exact for T_{max}=10,000s		Gap (%) for T_{max}=900s				Gap (%) for T_{max}=1,800s				Gap (%) for T_{max}=3,600s			
INSTANCE	#OPT	Cost	B	I	D	DI	B	I	D	DI	B	I	D	DI
40 4 0.80	4	684.69	0.42	0.41	0.64	0.29	0.60	0.40	0.62	0.50	0.27	0.26	0.27	0.25
40 4 0.82	5	688.40	0.12	0.08	0.12	0.64	0.11	0.53	0.00	0.41	0.27	0.54	-0.10	0.41
40 4 0.85	1	699.43	0.25	0.10	-0.08	-0.14	0.13	0.09	0.11	-0.14	-0.03	0.14	0.10	-0.22
50 3 0.85	5	677.33	0.17	0.23	0.10	0.24	0.15	0.16	0.14	0.13	0.09	0.14	0.14	0.11
50 3 0.88	4	685.36	0.35	0.62	0.52	0.35	0.38	0.65	0.51	0.50	0.53	0.79	0.43	0.06
50 3 0.90	1	671.28	0.58	0.64	0.40	0.34	0.43	0.62	0.42	0.44	0.42	0.62	0.51	0.64
50 4 0.80	5	741.10	0.44	1.10	0.65	1.12	0.42	0.90	0.34	0.81	0.33	0.75	0.20	0.30
50 4 0.82	1	751.49	0.08	0.23	0.08	0.35	0.11	0.20	0.15	0.20	0.08	0.20	0.06	0.17
50 4 0.85	1	745.02	1.10	0.64	0.98	0.66	0.90	0.54	0.90	0.45	0.21	-0.20	0.67	-0.19
60 2 0.90	10	675.80	0.00	0.02	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60 2 0.93	9	662.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60 2 0.95	4	677.86	0.00	0.03	0.00	0.02	0.02	0.02	0.00	0.02	0.00	0.02	0.02	0.02
60 3 0.85	6	714.89	0.00	0.15	0.05	0.05	-0.01	-0.02	0.23	0.21	0.10	0.19	0.02	0.16
60 3 0.88	2	738.69	0.01	0.02	0.04	0.05	0.03	0.02	-0.02	0.00	0.03	-0.01	0.00	0.02
60 3 0.90	1	723.39	0.13	0.07	0.21	0.07	0.04	0.01	0.07	-0.02	0.03	-0.02	0.13	-0.01
60 4 0.80	0	769.84	1.18	1.25	0.68	0.81	1.22	1.24	1.04	1.12	1.00	1.23	0.72	1.15
60 4 0.82	3	766.24	0.21	0.39	0.38	0.21	0.16	0.66	0.13	0.09	0.57	0.45	0.10	-0.11
60 4 0.85	0	784.43	0.19	0.26	0.37	0.25	0.27	0.24	0.24	0.10	0.22	0.07	-0.06	0.18
70 2 0.90	9	708.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
70 2 0.93	6	716.16	0.30	0.29	0.30	0.07	0.21	0.33	0.12	0.05	0.24	0.17	0.11	0.05
70 2 0.95	1	718.33	0.02	0.07	0.03	0.02	0.02	0.00	0.00	0.02	0.02	0.07	0.02	0.02
70 3 0.85	7	743.42	0.16	-0.03	0.04	-0.01	0.10	-0.01	0.16	-0.02	0.04	0.12	-0.03	-0.03
70 3 0.88	2	773.06	0.10	0.30	0.17	0.14	0.25	0.13	0.17	0.26	0.15	0.19	0.12	0.09
70 3 0.90	0	777.11	0.45	0.54	0.48	0.40	0.24	0.33	0.10	0.37	0.30	0.26	0.06	0.07
80 2 0.90	8	749.15	0.06	0.07	0.04	0.06	0.06	0.07	0.04	0.04	0.04	0.06	0.04	0.04
80 2 0.93	3	773.53	0.09	0.09	0.09	0.09	0.09	0.08	0.09	0.07	0.09	0.03	0.04	0.07
80 2 0.95	0	774.35	0.09	0.10	0.10	0.05	0.16	0.13	0.04	0.04	0.16	0.06	0.00	-0.02
AVG GAP			0.24	0.29	0.24	0.23	0.23	0.27	0.21	0.21	0.19	0.23	0.13	0.12

Table 4: Gaps (%) on all instances with $\kappa = 10$

	Exact for T_{max}=10,000s		Gap for T_{max}=900s				Gap for T_{max}=1,800s				Gap for T_{max}=3,600s			
INSTANCE	#OPT	Cost	B	I	D	DI	B	I	D	DI	B	I	D	DI
40 4 0.80	4	666.56	0.12	0.05	0.25	0.05	0.05	0.25	0.05	0.02	0.02	0.05	0.05	0.25
40 4 0.82	5	678.72	0.14	0.39	0.20	0.36	0.30	0.39	0.14	0.37	0.24	0.31	0.24	0.30
40 4 0.85	1	640.15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50 3 0.85	5	689.29	0.08	0.17	0.08	0.25	0.03	0.09	0.03	0.09	0.03	0.08	0.03	0.03
50 3 0.88	4	702.44	0.00	0.34	0.27	0.00	0.06	0.06	0.00	0.00	0.23	0.34	0.27	0.00
50 3 0.90	1	662.49	0.19	0.19	0.00	0.15	0.00	0.09	0.15	0.00	0.15	0.00	0.15	0.09
50 4 0.80	5	747.93	0.29	1.72	0.40	0.87	0.39	1.25	0.28	1.24	0.46	1.24	0.17	0.36
50 4 0.82	1	739.01	1.57	1.07	0.89	1.07	1.34	1.07	1.57	1.07	1.57	1.07	1.34	0.75
50 4 0.85	1	718.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60 2 0.90	10	675.81	0.00	0.02	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60 2 0.93	9	662.29	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60 2 0.95	4	680.51	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60 3 0.85	6	721.10	0.07	0.06	0.00	0.09	0.03	0.06	0.29	0.35	0.08	0.35	0.00	0.05
60 3 0.88	2	733.91	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60 3 0.90	1	702.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60 4 0.82	3	767.43	0.08	0.30	0.03	0.30	0.07	0.30	0.07	0.05	0.07	0.30	0.07	0.03
70 2 0.90	9	709.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
70 2 0.93	6	717.80	0.12	0.22	0.12	0.00	0.13	0.22	0.12	0.00	0.18	0.20	0.12	0.00
70 2 0.95	1	707.52	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
70 3 0.85	7	736.38	0.26	0.27	0.16	0.25	0.26	0.14	0.26	0.24	0.20	0.31	0.21	0.20
70 3 0.88	2	762.07	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
80 2 0.90	8	756.68	0.07	0.10	0.06	0.09	0.07	0.10	0.06	0.06	0.06	0.09	0.06	0.06
80 2 0.93	3	772.41	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AVG GAP			0.09	0.21	0.09	0.14	0.09	0.16	0.10	0.15	0.10	0.19	0.08	0.08

Table 5: Gaps (%) on instances solved to optimality with $\kappa = 10$

4.2 Performance analysis

In tables 6 and 7, we compare the performance of the matheuristics with the exact algorithm for $\kappa = 8$ and $\kappa = 10$, respectively. We recall that the exact algorithm was run for a maximum of 10,000 seconds. Considering the best obtained solutions, for each of the four matheuristics in combination with the three runtimes we report the following values:

- The number of instances where the exact algorithm outperformed the matheuristic (column two);
- The number of instances where the exact algorithm and the matheuristic obtained the same solutions (column three);
- The number of instances where the matheuristic outperformed the exact algorithm (column foure)

Tables 6 and 7 reconfirm that the combination of the diversification and the intensification procedures is superior to considering each procedure in separation. From Table 6 we note that matheuristic "DI" with runtime of 3,600 seconds obtained 63 new best solutions. For this combination of parameters, the matheuristic yields solutions that are equivalent or better than those obtained by the exact algorithm in 78.9% of all instances. From Table 7 we note that matheuristic "DI" with runtime of 3,600 seconds obtained 61 new best solutions. Moreover, the matheuristic "DI" with $\kappa = 8$ yields a higher number of new best solutions than matheuristic "DI" with $\kappa = 10$ in the three runtime categories.

Considering runtimes of 900 seconds, i.e., 9% of the time allotted to the exact algorithm, all four matheuristics with both $\kappa = 8$ and $\kappa = 10$ yield solutions that are equivalent or better than those obtained with the exact algorithm in at least 60% of all instances. With a runtime of 900 seconds "DI" with $\kappa = 8$ obtained 56 new best solutions.

5. Conclusions

We proposed a matheuristic for the VRPSD with classical recourse. Considering a multi-descent scheme in a local branching framework, we introduced an intensification procedure and a diversification procedure. Furthermore, we allowed non-improving solutions within a descent and proposed a dynamic computational time management procedure. We conducted extensive computational computational experiments to assess the effectiveness of the

900s	Exact < Local Branching	Exact = Local Branching	Exact > Local Branching
B	107	120	43
I	99	122	49
D	104	121	45
DI	78	136	56
1,800s	Exact < Local Branching	Exact = Local Branching	Exact > Local Branching
B	106	120	44
I	85	129	56
D	90	130	50
DI	66	144	60
3,600s	Exact < Local Branching	Exact = Local Branching	Exact > Local Branching
B	91	125	54
I	81	132	57
D	80	134	56
DI	57	150	63

Table 6: Number of instances with $\kappa = 8$

900 s	Exact < Local Branching	Exact = Local Branching	Exact > Local Branching
B	103	119	48
I	105	118	47
D	100	123	47
DI	91	127	52
1,800 s	Exact < Local Branching	Exact = Local Branching	Exact > Local Branching
B	101	123	46
I	95	123	52
D	94	125	51
DI	86	132	52
3,600s	Exact < Local Branching	Exact = Local Branching	Exact > Local Branching
B	93	122	55
I	84	127	59
D	85	129	56
DI	70	139	61

Table 7: Number of instances with $\kappa = 10$

matheuristic. The quality of the solutions were benchmarked against the best obtained solutions from an exact algorithm with a runtime of 10,000 seconds. All experiments pertaining to the matheuristic were run over 900, 1,800 and 3,600 seconds.

The combination of the intensification procedure and the diversification procedure performed substantially better than when these procedures were executed in separation. Considering the 270 instances, the combination of the intensification procedure and the diversification procedure with $\kappa = 8$ yielded 63 new best known solution within 3,600 seconds. Compared to the best obtained solutions from the exact algorithm, an average gap of 0.12% on all instances was achieved. Considering runtimes of 900 and 1,800 seconds, the matheuristic achieved 56 and 60 new best known solutions, respectively. The exact algorithm solved 98 instances to optimality within the allotted time. Considering these instances, the combination of the intensification procedure and the diversification procedure with $\kappa = 10$ yielded an average gap of 0.08% with a runtime limit of 3600 seconds.

Local branching heavily relies on the ability to efficiently solve restricted solution spaces of the problem. Therefore, the proposed matheuristic could be adapted to more involved recourse actions, once exact algorithms are developed for them. Further research could aim at defining new neighbourhood structures fitting the proposed local branching framework for the VRPSD.

References

[1] A. Ak and A. Erera. A paired-vehicle recourse strategy for the vehicle-routing problem with stochastic demands. *Transportation Science*, 41:222–237, 2007.

[2] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

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

[4] D. J. Bertsimas, P. Jaillet, and A. R. Odoni. A priori optimization. *Operations Research*, 38:1019–1033, 1999.

[5] D.J. Bertsimas. *Probabilistic Combinatorial Optimization Problems*. PhD thesis, Operations Research Center, Massachusetts Institute of Technology, 1988.

[6] K. Chepuri and T. Homem de Mello. Solving the vehicle routing problem with stochastic demands using the cross entropy method. *Annals of Operations Research*, 134:153–181, 2005.

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

[8] M. Fischetti and A. Lodi. Local branching. *Mathematical programming*, 98:23–47, 2003.

[9] C. Gauvin, M. Gendreau, and G. Desaulniers. A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands. *Computers & Operations Research*, 50:141–153, 2014.

[10] M. Gendreau, O. Jabali, and W. Rei. Stochastic vehicle routing problems. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, pages 213–240. MOS-SIAM series on Optimization, Philadelphia, 2014.

[11] M. Gendreau, O. Jabali, and W. Rei. Future research directions in stochastic vehicle routing. *Transportation science*, 2016. To appear.

[12] M. Gendreau, G. Laporte, and R. Séguin. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science*, 29:143–155, 1995.

[13] M. Gendreau, G. Laporte, and R. Séguin. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44:469–477, 1996.

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

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

[16] C. Hjorring and J. Holt. New optimality cuts for a single-vehicle stochastic routing problem. *Annals of Operations Research*, 86:569–584, 1999.

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

[18] P. Jaillet. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36:929–936, 1988.

[19] V. Lambert, G. Laporte, and F.V. Louveaux. Designing collection routes through bank branches. *Computers & Operations Research*, 20:783–791, 1993.

[20] G. Laporte and F. V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13:133–142, 1993.

[21] G. Laporte, F. V. Louveaux, and L. Van hamme. An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50:415–423, 2002.

[22] H. Lei, G. Laporte, and B. Guo. The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, 38:1775–1783, 2011.

[23] A. Leuliet. Nouvelles coupes pour le problème de tournées de véhicule avec demandes stochastiques. Master's thesis, École Polytechnique de Montréal, 2014.

[24] J.E. Mendoza, L.M. Rousseau, and J.G. Villegas. A hybrid metaheuristic for the vehicle routing problem with stochastic demand and duration constraints. *Journal of Heuristics*, 22:1–28, 2015.

[25] J.E. Mendoza and J.G. Villegas. A multi-space sampling heuristic for the vehicle routing problem with stochastic demands. *Optimization Letters*, 7:1503–1516, 2013.

[26] W. Rei, M. Gendreau, and P. Soriano. A hybrid monte carlo local branching algorithm for the single vehicle routing problem with stochastic demands. *Transportation Science*, 44(1):136–146, 2010.

[27] N. Secomandi and F. Margot. Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Operations Research*, 57:214–230, 2009.

[28] R. M. Van Slyke and R. Wets. *L*-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17:638–663, 1969.

[29] P. Toth and D. Vigo, editors. *Vehicle routing: problems, methods, and applications.* MOS-SIAM series on Optimization, Philadelphia, 2014.

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

Acknowledgments

The authors gratefully acknowledge funding provided by the Canadian Natural Sciences and Engineering Research Council.