A Hybrid Monte Carlo Local Branching Algorithm for the Single Vehicle Routing Problem with Stochastic Demands

Walter Rei
Michel Gendreau
Patrick Soriano

July 2007

CIRRELT-2007-24

A Hybrid Monte Carlo Local Branching Algorithm for the Single Vehicle Routing Problem with Stochastic Demands

Walter Rei[1,2,*], **Michel Gendreau**[1,3], **Patrick Soriano**[1,4]

1. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7
2. École des Sciences de la Gestion, Université du Québec à Montréal, 315 Ste-Catherine Est, Montréal, Canada H2X 3X2
3. Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7
4. Service de l'enseignement des méthodes quantitatives de gestion, HEC Montréal, 3000 Côte-Ste-Catherine, Montréal, Canada H3T 2A7

Abstract. We present a new algorithm that uses both local branching and Monte Carlo sampling in a multi-descent search strategy for solving 0-1 integer stochastic programming problems. This procedure is applied to the single vehicle routing problem with stochastic demands. Computational results show the usefulness of this new approach to solve hard instances of the problem.

Keywords. Local branching, Monte Carlo sampling, stochastic vehicle routing problems.

Acknowledgements. Financial support for this work was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Fonds québécois de la recherche sur la nature et les technologies (FQRNT). This support is gratefully acknowledged.

---

* Corresponding author: rei.walter@uqam.ca

This document is also published as Publication #1302 by the Department of Computer Science and Operations Research of the Université de Montréal.

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2007

1 Introduction

There has been a great deal of research done on vehicle routing problems (VRP). Both exact algorithms and metaheuristic procedures have been proposed for deterministic cases of the VRP (see [30]). However, in practice, one rarely has access to perfect information concerning the parameters of a problem. Therefore, in recent years, stochastic versions have been considered where certain parameters of the VRP are modeled by random variables. By solving stochastic routing problems, one can obtain significantly better solutions whenever there is uncertainty in the situation being modeled. These problems are therefore very interesting for real life applications but they are unfortunately notoriously hard to solve.

In this paper, the problem that is studied is the single vehicle routing problem with stochastic demands (SVRPSD). The SVRPSD is defined as follows: let $G(V, E)$ be an undirected graph, where $V = \{v_1, \ldots, v_N\}$ is a set of vertices and $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ is a set of edges. Defined on E is a symmetric matrix $C = [c_{ij}]$ that corresponds to the travel costs between vertices. Vertex v_1 represents a depot from which the vehicle must start and finish its route. If one searches for a route that visits all vertices once and minimizes the total travel cost, then one is in fact solving the well known travelling salesman problem (TSP). The TSP is an NP-hard problem which has been extensively studied, see [14].

The SVRPSD is obtained by adding a particular component to the classical TSP problem. Let us suppose that the vehicle has a limited capacity D and that each vertex $j \in V \setminus \{v_1\}$ corresponds to a customer that has a nonnegative demand ξ_j that is stochastic. Let us also make the following hypothesis: $\forall j \in V \setminus \{v_1\}$, demand ξ_j only becomes known when the vehicle arrives at the location of customer j. In this case, whenever a customer is visited, the residual capacity of the vehicle may not suffice to fulfill the observed demand. When such a failure occurs, one must take a recourse action that will entail an extra cost.

The model used here is based on the classical two stage stochastic programming formulation. In the first stage one constructs a route that visists all customers once. In the second stage, the determined route is followed and demands become known. When a failure occurs, partial delivery is performed and the recourse action taken is to return to the depot, to stock up (or to unload), and then go back to the customer where failure occured to finish the delivery and continue the route. In this case, the extra cost incurred is the traveling cost to the depot and the return cost to the customer location. The optimization problem consists of finding a route that minimizes the sum of both the total travel cost as well as the expected cost of recourse. For a complete description of the models that can be used as well as the properties associated with them, the reader is refered to the papers of Dror and *al.* [7, 6].

What makes the SVRPSD a hard problem to solve is the combination of both the inherent complexity of the TSP and the stochastic cost associated with the feasible solutions. If one defines the expected filling rate of the vehicle as: $\overline{f} = \sum_{j=1}^{N} \mathbf{E}[\xi_j]/D$, then as $\overline{f}$ increases so does the risk of failures. Instances where both the number of customers and $\overline{f}$ are large are very hard to solve optimaly (see [24]). The main contribution of this paper is to propose a new heuristic algorithm that solves efficiently such hard instances. The heuristic developped is a hybrid method that uses both local branching and Monte Carlo sampling. Certain characteristics of the SVRPSD will be exploited in the implementation of the method. However, the methodology that is proposed is quite general and can be applied to other stochastic problems.

The remainder of this paper is divided as follows. In section 2, the model used for the SVRPSD is presented as well as a brief description of the solution methods to solve it. In section 3, various Monte Carlo methods that have been developped to solve stochastic programming problems are reviewed. Section 4 includes a description of the heuristic that is proposed in this paper. This is followed by the computational results obtained on the SVRPSD in section 5. Finally, section 6 presents some concluding remarks.

2 The SVRPSD

The model for the SVRPSD is defined as follows:

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

$$\text{s.t.} \quad \sum_{j=2}^{N} x_{1j} = 2, \tag{2}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2, \; k = 2, \ldots, N, \tag{3}$$

$$\sum_{i \in S} \sum_{\substack{j \notin S \\ j>i}} x_{ij} + \sum_{i \notin S} \sum_{\substack{j \in S \\ j>i}} x_{ij} \geq 2, \; S \subseteq V, \; |S| \geq 3, \tag{4}$$

$$x_{ij} \in \{0,1\}, \; 1 \leq i < j \leq N. \tag{5}$$

Function $\mathcal{Q}(x)$ in (1) is the recourse function which represents the expected cost of recourse. It should be specified that under some assumptions, given a feasible route x, function $\mathcal{Q}(x)$ can be easily computed, as described in the paper of Laporte and *al.* [19]. Constraints (2) and (3) are used to make sure that the route starts and ends at the depot and that each customer is visited once. Inequalities (4) are the subtour elimination constraints. Finally, constraints (5) impose the integrality restriction on the variables of the problem.

The principal solution approach that is used to solve problem (1)-(5) is based on the 0-1 integer L-shaped algorithm presented by Laporte and Louveaux [18]. Following this approach, Benders decomposition is applied to the problem. The recourse function $\mathcal{Q}(x)$ is replaced in the objective by variable Θ which is then bounded by a series of optimality cuts (or any other lower bounding functionnals). In addition, constraints (4) and (5) are relaxed from the model. Optimality cuts as well as constraints (4) are then gradually added to the relaxed problem following a branch and cut framework.

Gendreau and *al.* [12] were the first to apply the standard L-shaped algorithm to the single vehicle routing problem with stochastic demands. In 1999, Hjorring and Holt [16] proposed a new type of cut that uses information taken from partial routes. A partial route is made up of three sets. Using the notation proposed by Laporte and *al.* [19], let us first define the two ordered sets $S = \{v_1, \dots, v_s\}$ and $T = \{v_1, \dots, v_t\}$. Sets S and T must respect the following condition: $S \cap T = \{v_1\}$. Let us now define a third set $U = V \setminus ((S \setminus \{v_s\}) \cup (T \setminus \{v_t\}))$. One easily sees that $S \cap U = \{v_s\}$ and $T \cap U = \{v_t\}$. Therefore, a partial route is made up of the two vectors $(v_1, \dots, v_s)$ and $(v_t, \dots, v_1)$ that define the beginning and end portions of the route and of set U, which contains all vertices that are not yet ordered. If $(v_i, v_j) \in S$ or T refers to the case where v_i and v_j are consecutive in S or T, then let $W(x) = \sum_{(v_i,v_j)\in S} x_{ij} + \sum_{(v_i,v_j)\in T} x_{ij} + \sum_{v_i,v_j\in U} x_{ij} - |V| + 1$. If Q is a lower bound on the value of recourse for the partial route and L is a general lower bound on $\mathcal{Q}(x)$, then the following inequality is valid for problem (1)-(5):

$$\Theta \geq L + (Q - L)W(x). \tag{6}$$

In [16], the authors present a lower bounding technique to obtain value Q. Laporte and *al.* [19] generalize (6) to the case of multiple vehicles. They also develop a new technique to obtain a better general lower bound L.

Recently, Rei and *al.* [24] proposed a new type of valid inequalities that apply to the case of the 0-1 integer L-shaped algorithm. These inequalities are based on local branching descents and are applied to problem (1)-(5). The implementation of the 0-1 integer L-shaped algorithm for the SVRPSD proposed in [24] produces the best results for the case where demands are Normal random variables (i.e., $\xi_j \sim N(\mu_j, \sigma_j)$) and where all random variables are independently distributed. However, instances where both the filling rate and the number of customers are large still present a tremendous challenge which justifies the development of efficient heuristics for this problem.

Heuristics have been proposed for related versions of problem (1)-(5). Gendreau and *al.* [13] proposed a tabu search algorithm for routing problems where customers and demands are stochastic. In 2000, Yang and *al.* [32] have proposed a series of heuristics for routing problems with stochastic demands for which restocking is considered. Restocking allows the vehicle to return to the depot before visiting the next customer on the route. By doing so, one may prevent failures. Recently, Bianchi and *al.* [2] have also implemented a series of metaheuristics for stochastic routing problems that allow restocking. Secomandi [26, 27] proposes neuro-dynamic programming algorithms for the case where reoptimization is applied to the SVRPSD. In this case, as demands become known, the ordering of the customers that have not yet been visited may be changed depending on the state of the situation. Finally, Chepuri and Hommem-De-Mello [4] solve an alternate formulation of the SVRPSD using the cross-entropy method. The alternate formulation considered allows the possibility that certain customers may not be serviced by the vehicle. However, a penalty function is used to dissuade such situations.

3 Monte Carlo sampling in stochastic programming

In this section a general presentation of how Monte Carlo sampling has been used in stochastic programming is provided. Note however that this section does not aim at being exhaustive but focuses on presenting the principal results and solution approaches within this field. Let us first define the classical stochastic programming problem with fixed recourse as follows:

$$\begin{aligned} \text{Min} \quad & c^\top x + \mathcal{Q}(x) && (7)\\ \text{s.t.} \quad & Ax = b && (8)\\ & x \in \overline{X}, && (9) \end{aligned}$$

where $\mathcal{Q}(x) = \mathbf{E}_\xi[Q(x, \xi(\omega))]$ and $Q(x,\xi(\omega)) = Min_y\{q(\omega)^\top y \mid Wy = h(\omega) - T(\omega)x,\ y \in \overline{Y}\}$, $\forall \omega \in \Omega$. Monte Carlo sampling is mainly used in two different ways to solve problem (7)-(9). As presented by Linderoth and *al.* [20] sampling is either used in an interior fashion or in an exterior fashion. When sampling is used in an interior fashion, one is actually trying to solve directly problem (7)-(9) but whenever the algorithm being used requires information concerning the recourse function, then sampling is applied to approximate this information. In the exterior approach, instead of trying to solve the stochastic problem directly, one uses sampling beforehand as a way to approximate the recourse function. One can then apply any adapted deterministic optimization algorithm to solve the approximated problem.

The first type of methods that have been proposed using the interior approach are based on the L-shaped algorithm presented by Van Slyke and Wets [29] for the case of continuous stochastic programming problems with fixed recourse. The first to introduce sampling in the L-shaped algorithm were Dantzig and Glynn [5]. The algorithm proposed in [5] uses sampling to estimate the cuts needed in the solution process. Samples are determined so as to obtain a given confidence level. To improve the convergence rate, importance sampling is used for the generation of the scenarios. Since the size of the samples needed can become quite large, the authors also propose a parallel implementation of the method to reduce solution times.

Another method that uses sampling in an L-shaped based algorithm is the stochastic decomposition approach proposed by Higle and Sen [15] for the case of problems with complete recourse (i.e., $Q(x, \xi(\omega)) < \infty$ regardless of x and $\forall \omega \in \Omega$). The idea behind stochastic decomposition is to use larger samples to produce cuts as the number of iterations of the L-shaped algorithm increases. At iteration ν, the algorithm uses ν independently generated samples to produce the next optimality cut. Previously generated cuts are updated in such a way that they become redundant and are subsequently dropped as the algorithm proceeds. Details concerning the convergence and implementation of this approach are provided in [15]. The authors also elaborate on the use of stopping rules, which include both error bound estimates and tests on optimality conditions.

Finally, stochastic quasi-gradient methods, see Ermoliev [8], have also applied Monte Carlo sampling in an interior fashion. In this case, sampling is used to produce a subgradient or quasi-gradient for which a descent direction may be obtained. The algorithm proceeds by taking a step in the direction that is defined. A projection is then applied onto the set of feasible first stage solutions.

Techniques that use Monte Carlo sampling in an exterior fashion are generally based on the use of sample average approximations of the recourse function. Let $X = \{x \mid Ax = b,\ x \in \overline{X}\}$ be the set of first stage constraints, then one may rewrite problem (7)-(9) as: $\min\limits_{x \in X} f(x)$ where $f(x) = \mathbf{E}_\xi[c^\top x + Q(x, \xi(\omega))] = c^\top x + \mathbf{E}_\xi[Q(x, \xi(\omega))]$. If $\{\omega^1, \ldots, \omega^n\}$ is a subset of randomly generated events of Ω, then function $\widehat{f}_n(x) = c^\top x + \frac{1}{n} \sum\limits_{i=1}^{n} Q(x, \xi(\omega^i))$ is a sample average approximation of $f(x)$. One may now define the approximating problem in the following way: $\min\limits_{x \in X} \widehat{f}_n(x)$.

It is shown in Mak and *al.* [23] that if one considers the average value of the approximating problem over all possible samples, then one obtains a lower bound on the optimal value of problem (7)-(9), that is: $\mathbf{E}\left[\min\limits_{x \in X} \widehat{f}_n(x)\right] \leq \min\limits_{x \in X} f(x)$. In [23], the same type of reasoning is also applied to the case where one is trying to compute the value of a first stage feasible solution. Let $\tilde{x}$ be a feasible first stage solution, then one may show that $\mathbf{E}\left[\widehat{f}_n(\tilde{x})\right] \geq f(\tilde{x})$. Therefore, by using unbiased estimators for $\mathbf{E}\left[\min\limits_{x \in X} \widehat{f}_n(x)\right]$ and for $\mathbf{E}\left[\widehat{f}_n(\tilde{x})\right]$, one can construct confidence intervals on the optimal gap associated with solution $\tilde{x}$. Unbiased estimators can be obtained by using batches of subsets $\{\omega^1, \ldots, \omega^n\}$. Let $\widehat{f}_n^j$

be the jth sample average approximation function using a randomly generated subset of size n and let $\widehat{v}_n^j = \min_{x \in X} \widehat{f}_n^j(x)$, for $j = 1, \ldots, m$. Then $L_m^n = \frac{1}{m} \sum_{j=1}^{m} \widehat{v}_n^j$ and $U_m^n = \frac{1}{m} \sum_{j=1}^{m} \widehat{f}_n^j(\tilde{x})$ can be used to estimate the gap associated with $\tilde{x}$. In [23], some variance reduction techniques are also presented.

Under certain conditions, if $\hat{x}_n$ is an optimal solution to problem $\min_{x \in X} \widehat{f}_n(x)$, then it can be shown that $\hat{x}_n$ converges with probability 1 to the set of optimal solutions to (7)-(9) as $n \to \infty$. Furthermore, when the probability distribution of ξ is discrete, given some assumptions, Shapiro and Homem-De-Mello [28] show that $\hat{x}_n$ is an exact optimal solution to (7)-(9) for n large enough. The authors also demonstrate that the probability associated with the event of $\hat{x}_n$ not being an optimal solution to (7)-(9) tends to zero exponentially fast as $n \to \infty$. Using these results, Kleywegt and *al.* [17] elaborate the sample average approximation (or SAA) method.

The SAA method randomly generates batches of samples of random events and then solves the approximating problems. Each solution obtained is an approximation of the optimal solution to the original stochastic problem. Estimates on the optimal gap using bounds L_m^n and U_m^n are then generated to obtain a stopping criteria. Value n may be increased if either the gap or the variance of the gap estimator is to large. In [17], the authors also discuss the use of postprocessing procedures that provide some guarantees as to the quality of the solution chosen by the algorithm. The SAA method was adapted for the case of stochastic programs with integer recourse by Ahmed and Shapiro [1]. Recently, Linderoth and *al.* [20] have produced a series of numerical experiments using the SAA method which show the usefulness of the approach.

4 Monte Carlo local branching hybrid algorithm

The SAA algorithm has been successfully applied to obtain good quality solutions for a variety of stochastic problems for which direct solution approaches are inefficient (see [31], [25] and [20]). However, one is not always able to solve efficiently the approximating problems needed for the SAA approach. This situation has been observed in the case of hard istances of the SVRPSD. In this section, a heuristic that uses both local branching and Monte Carlo sampling will be presented to obtain good quality solutions even if the approximating problems obtained after sampling are still too difficult to solve in a reasonable time. This section will be divided in two subsections: the first will include a presentation of the local branching methodology, in the second subsection, a description of the solution approach using Monte Carlo sampling and local branching will be provided.

4.1 Local branching

The local branching solution approach was introduced by Fischetti and Lodi [9] as a way to solve hard mixed integer problems. The idea behind this method is to take advantage of the efficiency of generic solvers, such as CPLEX, for solving small integer 0-1 problems. Therefore, one can divide the feasible space of a problem into a series of smaller subregions and then use a generic solver to explore each of the subregions thus created.

To better illustrate this approach, let us apply local branching to the case of problem (7)-(9). To do so, let us first consider that problem (7)-(9) has binary first stage variables. Let us also suppose that the stochastic problems to be solved are such that all feasible first stage solutions are also feasible in the second stage (i.e., relative complete recourse). In this case, if vector x is of size n_1, then the set of first stage constraints may be defined in the following way: $X = \{x \mid Ax = b,\ x \in \overline{X} \cap \{0,1\}^{n_1}\}$. Again, if $f(x) = c^\top x + \mathcal{Q}(x)$, then problem (7)-(9) becomes:

$$\text{Min} \quad f(x) \tag{10}$$
$$\text{s.t.} \quad x \in X. \tag{11}$$

Let x^0 be a vector of 0-1 values such that $x^0 \in X$. Using x^0, let function: $\Delta(x, x^0) = \sum\limits_{j \in S_0} (1 - x_j) + \sum\limits_{j \in N_1 \setminus S_0} x_j$, where $N_1 = \{1, \ldots, n_1\}$ and $S_0 = \{j \in N_1 \mid x_j^0 = 1\}$, define the Hamming distance relative to x^0. Using function $\Delta(x, x^0)$ and a fixed integer value κ, one may divide problem (10)-(11) into two subproblems: the first having first stage feasible region $\{x \mid x \in X,\ \Delta(x, x^0) \leq \kappa\}$ and the second having $\{x \mid x \in X,\ \Delta(x, x^0) \geq \kappa + 1\}$. When κ is fixed to an appropriate (small) value, constraint $\Delta(x, x^0) \leq \kappa$ can considerably reduce the size of the feasible region of problem (10)-(11). Therefore, one can use an adapted generic solver to solve this subproblem efficiently. The subregion defined by $\Delta(x, x^0) \geq \kappa + 1$ is left for further exploration.

Let us now consider two finite index sets I^ν and J^ν such that $x^k \in X$, $\forall k \in I^\nu \cup J^\nu$. If x^ν is a feasible first stage solution such that $\nu \notin I^\nu$ and κ_i, $\forall i \in I^\nu \cup \{\nu\}$, is a series of fixed integer values, then let us define the following two subproblems:

$$\begin{array}{llc} (P_\nu) & Min. & f(x) \\ & \text{s.t} & \Delta(x, x^j) \geq 1, j \in J^\nu \\ & & \Delta(x, x^i) \geq \kappa_i, i \in I^\nu \\ & & \Delta(x, x^\nu) \leq \kappa \\ & & x \in X \end{array} \qquad \begin{array}{llc} (\overline{P}_\nu) & Min. & f(x) \\ & \text{s.t} & \Delta(x, x^j) \geq 1, j \in J^\nu \\ & & \Delta(x, x^i) \geq \kappa_i, i \in I^\nu \\ & & \Delta(x, x^\nu) \geq \kappa + 1 \\ & & x \in X. \end{array}$$

The local branching algorithm proceeds by solving subproblem P_ν using the generic solver. Subproblem P_ν is either feasible, in which case one obtains a solution $x^{\nu+1}$, or infeasible. If one obtains $x^{\nu+1}$, then either $f(x^{\nu+1}) < f(x^{\nu})$ or $f(x^{\nu+1}) \geq f(x^{\nu})$. If $f(x^{\nu+1}) < f(x^{\nu})$ then the algorithm sets $\kappa_{\nu} = \kappa + 1$, $I^{\nu+1} = I^{\nu} \cup \{\nu\}$ and $J^{\nu+1} = J^{\nu}$. Constraint $\Delta(x, x^{\nu}) \leq \kappa$ is replaced by $\Delta(x, x^{\nu}) \geq \kappa_{\nu}$, which gives us subproblem $\overline{P}_{\nu}$. Following the same branching scheme, solution $x^{\nu+1}$ is then used to separate the feasible region of $\overline{P}_{\nu}$, thus creating subproblems $P_{\nu+1}$ and $\overline{P}_{\nu+1}$. At this point, $P_{\nu+1}$ becomes the next subproblem to be solved. In the case where $f(x^{\nu+1}) \geq f(x^{\nu})$ or P_{ν} is infeasible, a diversification procedure is applied. The diversification procedure follows the principle that in order to obtain a better solution (or a feasible subproblem), then the feasible region of P_{ν} must be increased. Therefore, if $f(x^{\nu+1}) \geq f(x^{\nu})$, then constraint $\Delta(x, x^{\nu+1}) \geq 1$ is added to the subproblem and $J^{\nu+1} = J^{\nu} \cup \{\nu + 1\}$. By doing so, one eliminates from further consideration a solution $x^{\nu+1}$ whose value is no better than that of x^{ν}. In order to increase the size of the current subproblem feasible region, constraint $\Delta(x, x^{\nu}) \leq \kappa$ is replaced by $\Delta(x, x^{\nu}) \leq \kappa + \lceil \frac{\kappa}{2} \rceil$. By fixing $I^{\nu+1} = I^{\nu}$, one obtains $P_{\nu+1}$, which will be the next subproblem to be solved in the search process.

It should be specified that the branching decision may be applied using a different criteria then the one that has been evoked. Furthermore, for the diversification strategy, one may also use a different increase in the update of constraint $\Delta(x, x^{\nu}) \leq \kappa$. Local branching offers a general search context that one may adapt to the type of problem being solved. In [9], the authors impose a time limit for the solution of the subproblems. A series of diversification mechanisms derived from local search metaheuristics are also proposed. For the purpose of this paper, we will simply define a local branching descent as being a series of subproblems P_0, P_1, ..., that are solved to optimality or until a specified time limit is reached. The structure of each descent will be described in the next subsection.

4.2 Monte Carlo sampling and local branching

Monte Carlo sampling can be used to approximate the recourse function in (7)-(9). In doing so, one alleviates the stochastic complexity of the problem. By using local branching to explore X, one is able to control the combinatorial complexity associated with the first stage of problem (7)-(9). We will now show how these strategies may be used in a coordinated fashion creating what we will refer to as a multi-descent algorithm for the SVRPSD. To do so, we will first explain the multi-descent scheme and then describe the local branching descent structure used.

4.2.1 Multi-descent scheme

For the moment, let us suppose that one is able to solve efficiently local branching subproblems whitout resorting to sampling. Let us also consider the mean value problem (MVP), or expected value problem as in [3], associated with (7)-(9). The MVP problem is obtained by taking the random parameters of the stochastic problem and replacing them by their mean values. Using the formulation introduced in the previous sections, the MVP problem can be stated as follows:

$$\text{Min} \quad c^\top x + Q(x, \overline{\xi}) \tag{12}$$

$$\text{s.t.} \quad x \in X, \tag{13}$$

where $\overline{\xi} = \mathbf{E}[\xi]$. If $\overline{x}$ is an optimal solution to (12)-(13) and $x^\star$ is an optimal solution to the stochastic problem (10)-(11), then it is a well known result that $f(x^\star) \leq f(\overline{x})$ (see [3]). Actually, $f(\overline{x}) - f(x^\star)$ defines the value of the stochastic solution (VSS), which can be arbitrarily small or large depending on the problem considered.

In the case of routing problems, Louveaux [21] showed the importance of using the stochastic formulation when one considers the VSS. Although the VSS may be large, there is an important point to be made concerning the relative weight of the first stage objective function $c^\top x$ versus the recourse function $\mathcal{Q}(x)$ in the case of the SVRPSD. A problem where the expected filling rate $\overline{f}$ is small is in general easier to solve because it resembles the TSP. In this case, the MVP (12)-(13), or simply the TSP ($\min\limits_{x \in X} c^\top x$), offers a good approximation for the original problem (10)-(11). As $\overline{f}$ increases so does the risk of failures and $\overline{x}$ becomes a potentially bad route when considering the stochastic formulation. However, $x^\star$ usually remains a good solution for the MVP (or the TSP). The reason for this is that, with the exception of extreme cases, the travel cost of a route ($c^\top x$) generally outweighs the value of recourse ($\mathcal{Q}(x)$). Therefore, a route for which the travel cost is high will unlikely be optimal even if the recourse value is small. The main idea behind the algorithm proposed in this paper will be to use as starting point solution $\overline{x}$, or any other route whose travel cost is low, and then try to close the gap to obtain $x^\star$.

If one defines x^0 as a feasible solution to the stochastic problem, then let P_0^k, P_1^k, ... , $P_{l_k}^k$ be the kth finite local branching descent starting from x^0. Let us suppose that only solution x^0 is eliminated using the Hamming distance function in problem (10)-(11). In this case, the last subproblem solved (i.e. $P_{l_k}^k$)

is:

$$\begin{aligned}
\text{Min} \quad & f(x) && (14)\\
\text{s.t.} \quad & \Delta(x, x^0) \geq 1 && (15)\\
& \Delta(x, x^i) \geq \kappa_i, i \in I^{l_k} && (16)\\
& \Delta(x, x^{l_k}) \leq \kappa_{l_k} && (17)\\
& x \in X. && (18)
\end{aligned}$$

Since a different solution is used each time the branching decision is taken in a local branching descent, then from $P_0^k, \ldots, P_{l_k}^k$, one obtains at least l_k different first stage solutions.

From a multi-descent point of vue, one needs a feasible first stage solution to start a new descent. If one considers the kth local branching descent, then solutions $x^1, \ldots, x^{l_k}$, are all possible starting points. Using $x^1, \ldots, x^{l_k}$, the strategy that was chosen was to identify the best solution found in the last descent, $x^{k^\star} \in \arg\min\{f(x^i) \mid i = 1, \ldots, l_k\}$, add constraint $\Delta(x, x^{k^\star}) \geq 1$ to problem (10)-(11) and then use $x^{k^\star}$ as the new starting point of descent $k+1$. It should be noted that, in the case of the SVRPSD, $x^{k^\star}$ represents a feasible route. Therefore, constraint $\Delta(x, x^{k^\star}) \geq 1$ may be replaced by $\Delta(x, x^{k^\star}) \geq 4$, since all other feasible routes lie at a Hamming distance of at least four from $x^{k^\star}$ (see [24]). This type of descent will be refered to as a base descent. Since constraint $\Delta(x, x^{k^\star}) \geq 1$ is added to problem (10)-(11), the best solution that one finds in the $k+1$th descent will be different from the ones obtained in the previous k descents. Furthermore, since only solution $x^{k^\star}$ is eliminated, the algorithm can always come back and explore similar neighbourhoods from descent k to $k+1$. Base descents enable the algorithm to intensify the search around solutions that are found to be locally good. The drawback of this strategy is that if one eliminates all good feasible solutions in a certain vicinity of X or if one is exploring uninteresting neighbourhoods, then by only applying base descents, the procedure can take too long to reach different subregions.

To counter this potential problem, another strategy, using the local branching constraints, was applied in the multi-descent approach. If one considers descent k, then one may be satisfied by the extent of the exploration carried out in the subregions defined by subproblems $P_0^k, \ldots, P_{l_k}^k$. If one sets $I^{l_k+1} = I^{l_k} \cup \{l_k\}$ and $\kappa_{l_k} = \kappa_{l_k} + 1$, then one may be interested in applying the next local branching descent from a first stage solution defined by: $x \in \{x \in X \mid \Delta(x, x^i) \geq \kappa_i, \forall i \in I^{l_k+1}\}$. This corresponds to applying the next descent from a feasible solution to subproblem $\overline{P}_{l_k}^k$ associated with (14)-(18). To obtain this solution, one can add constraints $\Delta(x, x^i) \geq \kappa_i$, $\forall i \in I^{l_k+1}$, to the MVP (12)-(13) and then use the optimal solution to this new problem as a starting point for the $k+1$th descent. In the case of the SVRPSD, to make the search for this new solution easier, function $Q(x, \overline{\xi})$ is dropped from the objective and the problem that is used is the TSP. This amounts to starting the next local branching descent from a route whose travel cost is low but which is in a region that has not yet been explored.

Therefore, a meta phase will be defined as a series of local branching descents whose strating point will be an optimal solution (defined by x^{k+1}) to the following problem:

$$\begin{aligned} \text{Min} \quad & c^\top x && (19)\\ \text{s.t.} \quad & \Delta(x, x^i) \geq \kappa_i, \; i \in I^{l_j+1}, \; j = 1, \ldots, k && (20)\\ & x \in X. && (21) \end{aligned}$$

It should be specified that an optimal solution to (19)-(21) is not necessarily needed. A good feasible solution can be sufficient. The feasible first stage solution x^{k+1} does not lie in any of the neighbourhoods explored in the previous k descents. Constraint $\Delta(x, x^{k+1}) \geq 1$ is added to problem (10)-(11), and the next series of descents is executed. Meta phases provide a diversification strategy in the multi-descent scheme. It should be noted that the local branching constraints are only used in order to find a new starting point. They are not used in the following descents. This will allow the algorithm to come back to subregions which have already been visited if the solutions found are locally good.

4.2.2 Descent structure

One should now examine how the local branching descents may be performed. Since local branching subproblems of type P_ν may be hard to solve efficiently, sampling will be used. There is an important point to be made concerning the size of the samples that one may use for this approximation. Since the local branching search strategy is aimed at controling the complexity associated with the first stage of problem (7)-(9), then one may use larger samples in the approximation of the recourse function for P_ν compared to the original stochastic problem (10)-(11).

The original branching decision, in a local branching descent, is taken on the basis of the objective value of the solutions considered. When sampling is used, the information provided by the objective function of the approximated subproblems will no longer be completely accurate. Furthermore, if the feasible region of a subproblem becomes too large, it may turn out to be impossible to solve it efficiently. Therefore, the strategy that is used in this paper, is to keep value κ fixed and apply the branching decision each time a new solution is found. A descent will include a fixed number of levels, where each level will be comprised of a series of subproblems that are approximated using the same sample of random events.

We will now briefly describe the algorithm that will be used to solve the local branching subproblems. Each subproblem will be solved to optimality or until a specified time limit is reached. The procedure used will be the branch and cut algorithm presented by Rei and *al.* [24]. There are three types of cuts that are generated by the algorithm: subtour elimination constraints (4), partial route cuts (6) and local branching valid inequalities as defined in [24]. Constraints (4) are obtained by using the procedures in the CVRPSEP package proposed by Lysgaard and *al.* [22]. These constraints are valid for all local branching subproblems explored by the algorithm. Therefore, a pool of cuts will be defined in order to reuse previously identified constraints. Both partial route cuts and local branching valid inequalities use information on the recourse function. These cuts are therefore only valid for subproblems in the same level of a descent, that is, when the recourse function is approximated using the same sample. Partial route cuts will be reused on all subproblems in a given level. However, following the results obtained in [24], the local branching valid inequalities will be generated locally for each subproblem since this strategy was found to be more efficient.

In a given local branching descent, let $\{\omega_1^p, \ldots, \omega_n^p\}$ be the pth subset of randomly generated events of Ω for $p = 1, \ldots, m$, where m is the number of levels in the descent. If each level is made up of q subproblems ($P_{\nu p}$, $\nu = 1, \ldots, q$), then the local branching descent produces $m \times q$ different solutions. At the end of a descent, one must identify the best solution obtained. If x^i, $i = 1, \ldots, m \times q$, are the feasible solutions found, then one may use the m batches of randomly generated subsets to estimate the objective value of each solution. Therefore, let $\hat{f}_n^p(x) = c^\top x + \frac{1}{n}\sum_{j=1}^{n} Q(x, \xi(\omega_j^p))$ and $U_m^n(x) = \frac{1}{m}\sum_{p=1}^{m} \hat{f}_n^p(x)$, then the best solution obtained in iteration k will be estimated as being $x^{k^\star} \in \arg\min\{U_m^n(x^i) \mid i = 1, \ldots, m \times q\}$. This criterion directly follows the general principle that states that it is usually easier to find an ordering of the solutions rather than to correctly estimate their recourse value, see Fu [11].

For each local branching descent, one obtains a different feasible first stage solution that is identified as being the best one found in the neighbourhoods explored. Each of these solutions are obtained using different samples of random events. When the search process ends, one is left with the problem of having to identify the best solution found. Using simulation, a variety of methods have been proposed to deal with the problem of selecting between a finite number of possibilities. Following the classification provided by Fu [10], these methods can use the principles of multiple comparisons or ranking and selection. In the case of the SVRPSD, since one is now interested in evaluating the best recourse value obtained, then $\mathcal{Q}(x)$ will be measured for these solutions.

5 Computational results

In order to assess the performance of the proposed algorithm, we selected a subset of instances from the paper of Rei and *al.* [24]. These instances were choosen to be the problems of size $N = 60$, 70, 80 and 90 for which $\overline{f} = 1.025$, 1.05, 1.075 and 1.10 and that were classified as being hard to solve optimaly by the classical L-shaped algorithm (see [24]). We thus obtained a total of 60 instances. All results reported will be averages over the number of instances (nb. i.) for all values of N and $\overline{f}$. Also, a time limit of 60, 120, 180 and 240 seconds was imposed respectively on the solution process for the local branching subproblems for the instances of size $N = 60$, 70, 80 and 90.

Tests will be conducted in three phases. In the first phase, we will establish the appropriate structure of the local branching descents. This will include finding the best value for the size of the neighbourhoods (parameter κ) as well as the number of scenarios (parameter n) that should be used in order to solve the local branching subproblems. We will also determine the number of subproblems (parameter q) that should be included in each level of a descent. In the second phase, we will look at how results vary when the number of base descents is changed in the meta phases. Finally, we will analyse the quality of the results obtained by the best strategy for the multi-descent local branching algorithm. To do so, we will compare it to the L-shaped algorithm proposed in [24] for which a large time limit of 6000 seconds is imposed. It should be specified that all results for the heuristic algorithm are average values over five runs. Also, all experiments were performed on a 2.4 GHz AMD Opteron 64 bit processor.

			$\kappa = 4$			$\kappa = 6$			$\kappa = 8$		
N	$\overline{f}$	nb. i.	$n = 100$	$n = 200$	$n = 300$	$n = 100$	$n = 200$	$n = 300$	$n = 100$	$n = 200$	$n = 300$
60	1.025	1	1314.54	1313.2	1312.43∗†	1314.72	1313.72	1312.43∗†	1314.32	1312.43∗†	1312.43∗†
	1.050	3	1347.5	1344.68∗	1346.85	1344.34	1343.24∗	1345.85	1341.08∗†	1343.21	1343.47
	1.075	5	1333.97	1334.05	1333.74∗	1332.26	1332.28	1331.99∗†	1332.99	1333.06	1332.88∗
	1.100	5	1343.14	1343.13	1341.96∗	1338.03	1337.96∗†	1339.03	1341.01	1340.2∗	1340.41
70	1.025	3	1434.6∗	1434.72	1434.82	1430.89∗†	1433.67	1433.53	1433.25	1433.13	1432.38∗
	1.050	3	1401.91	1401.51	1401.33∗	1401.8	1401.11∗†	1401.8	1401.74	1401.3	1401.19∗
	1.075	5	1455.5	1454.82∗	1454.82∗	1444.4	1442.78∗	1445.72	1443.68	1442.68∗†	1443.89
	1.100	4	1498.74	1497.45∗	1499.62	1495.57	1498.28	1495.25∗	1496.08	1494.53∗†	1495.1
80	1.025	2	1483.34	1481.99∗†	1485.77	1482.98∗	1483.06	1483.17	1483.09	1482.94∗	1483.92
	1.050	2	1494.04∗	1494.43	1494.6	1494.18	1494.03	1493.93∗†	1494.72	1493.96	1493.93∗†
	1.075	5	1491.76∗	1491.87	1491.95	1487.55∗	1488.32	1488.13	1487.23	1486.55∗†	1487
	1.100	5	1503.97	1501.01	1500.12∗	1495.02	1494.89	1494.09∗†	1494.19∗	1494.36	1496.33
90	1.025	2	1576.52	1577.63	1575.63∗	1575.27∗	1575.31	1575.63	1574.74	1573.96	1573.08∗†
	1.050	5	1606.11	1606.29	1605.87∗	1605.56∗	1606.56	1606.45	1604.73∗†	1604.96	1605.58
	1.075	5	1605.54∗	1606.28	1605.92	1604.48	1604.62	1603.92∗	1602.51	1602.01∗†	1604.73
	1.100	5	1598.81∗	1599.23	1599.41	1599.02	1598.76	1598.59∗	1596.54∗†	1596.55	1596.71
Local best (∗)			5	4	8	5	4	7	4	7	6
Absolute best (†)			0	1	1	1	2	4	3	5	3

Table 1: Results: parameters κ and n

To establish the appropriate size of the neighbourhood as well as the number of scenarios necessary, the heuristic algorithm is first applied to produce one descent, starting from the solution to the TSP, where the number of levels is six ($m = 6$) and the number of subproblems for each level is one ($q = 1$). By doing so, each run performed includes a total of six subproblems solved. By fixing m

and q, one can better see the tradeoffs between both parameters κ and n. In Table 1, results are reported for the following values: κ = 4, 6 and 8 and n = 100, 200 and 300. Since in all cases the descent is performed using the same starting point, the algorithm will tend to search the same region of the problem. Therefore, one first observes that the differences in the quality of the solutions obtained are very small. Results in Table 1 also include the number of times each run obtained the best solutions for a given κ (Local best ($*$)), and, overall values of κ (Absolute best (†)). By analyzing these results, one is better able to distinguish which of the parameter settings produced the best local search. For any given value κ, one observes that larger values of n produce the best solutions in general. For κ = 4 and 6, n = 300 is best and as for κ = 8, n = 200 seems to be slightly better than n = 300. Including all runs, when one compares the number of times each value of κ obtains the best overall values then one can see that κ = 4 is best on a total of two occasions, κ = 6 on seven occasions and κ = 8 on 11 occasions. Therefore, it seems that by fixing κ = 8, one obtains the best local search. Furthermore, since one generally obtains better results for larger values of n, then for all following runs we will set: κ = 8 and n = 200.

N	$\overline{f}$	nb. i.	6-1	3-2	2-3	1-6
60	1.025	1	1312.43$*$	1313.00	1313.00	1313.02
	1.050	3	1343.21	1342.11$*$	1343.26	1343.33
	1.075	5	1333.06	1333.31	1333.02	1332.43$*$
	1.100	5	1340.2$*$	1340.99	1342.79	1340.47
70	1.025	3	1433.13	1432.72$*$	1433.49	1433.16
	1.050	3	1401.3	1401.54	1401.52	1400.83$*$
	1.075	5	1442.68$*$	1443.71	1443.92	1443.48
	1.100	4	1494.53$*$	1494.96	1494.85	1495.59
80	1.025	2	1482.94$*$	1483.41	1484.59	1483.79
	1.050	2	1493.96	1494.08	1493.92$*$	1494.10
	1.075	5	1486.55	1485.84$*$	1486.76	1487.03
	1.100	5	1494.36$*$	1494.40	1493.58	1493.99
90	1.025	2	1573.96$*$	1574.40	1575.21	1575.21
	1.050	5	1604.96$*$	1606.05	1606.60	1605.86
	1.075	5	1602.01	1600.31$*$	1603.53	1602.64
	1.100	5	1596.55	1596.69	1596.29	1595.17$*$
	Best ($*$)		7	4	2	3

Table 2: Results: parameters m and q

We will now examine how the concept of levels influences the quality of the results obtained. As was previously mentioned, solving several subproblems within a given level is interesting since the branch and cut algorithm may reuse the partial route cuts on all subproblems that are created using the same sample. In turn, this will accelerate the solution process for all subproblems on a given level. However, by reusing the same samples, one may also limit the search process. When solving a local branching subproblem, the quality of the solution obtained is dependent on the sample used to approximate the recourse function.

Results will be poor if a non representative sample is used. If the same non representative sample is applied on different subproblems then one can seriously limit the search process in the neighbourhoods that are explored.

In Table 2 the quality of the solutions obtained are presented when $m = 6$, 3, 2 and 1 and $q = 1$, 2, 3 and 6. For example, 6-1 will refer to the case where the total number of levels is six ($m = 6$) and each level contains one subproblem ($q = 1$). For these tests the number of descents is again limited to one, for a total of six subproblems solved in each run. By using the same starting point for each descent and by fixing both $\kappa = 8$ and $n = 200$, one is able to clearly see how results vary with the structure of the descent. The total number of times each run obtained the best results is also reported (Best ($*$)). Results in Table 2 seem to indicate that 6-1 outperforms all others. One obtains the best results on seven occasions with 6-1, compared to four for 3-2, two for 2-3 and three occasions for 1-6. It would seem that by using different samples for each subproblem, one better hedges against the risk of relying heavily on non representative samples. Therefore, the type of descent that will be used in the multi-descent approach will be made up of levels for which $q = 1$ and where subproblems are created using $\kappa = 8$ and $n = 200$.

N	$\overline{f}$	nb. i.	6/1	3/2	2/3	1/6
60	1.025	1	1310.94$*$	1312.43	1311.35	1312.07
	1.050	3	1342.24	1341.70	1341.08	1340.85$*$
	1.075	5	1332.04	1333.25	1331.92$*$	1331.92$*$
	1.100	5	1336.14$*$	1336.23	1336.26	1336.73
70	1.025	3	1430.79$*$	1431.99	1431.44	1432.39
	1.050	3	1399.65	1399.53$*$	1400.70	1401.14
	1.075	5	1440.37$*$	1441.03	1441.57	1442.53
	1.100	4	1492.70	1492.69$*$	1493.01	1493.62
80	1.025	2	1481.49	1481.04$*$	1481.94	1481.49
	1.050	2	1493.92$*$	1493.92$*$	1493.92$*$	1493.92$*$
	1.075	5	1483.76$*$	1485.43	1485.14	1485.70
	1.100	5	1489.91	1489.77$*$	1490.56	1489.95
90	1.025	2	1572.23	1571.98$*$	1572.11	1573.23
	1.050	5	1603.24$*$	1603.50	1604.63	1604.06
	1.075	5	1597.96$*$	1600.70	1598.80	1599.63
	1.100	5	1594.94	1594.84	1594.47	1594.06$*$
	Best ($*$)		8	6	2	4

Table 3: Results: Meta phases/Base descents

We will now examine the multi-descent search strategies that one may use for a given number of overall base descents. We fix the total number of base descents to six and each local branching descent performed is limited to a depth of three levels ($m = 3$), which produces a total of 18 subproblems solved by the algorithm. In Table 3, results are reported for the cases where the number of base descents varies in each meta phase. Therefore, the 6/1 column refers to the case where six meta phases of size one are performed. The 3/2 column represents runs made up of three meta phases of size two. The 2/3 column is for the case where two meta phases of size three are carried out. Finally, the 1/6 column is the case where six consecutive base descents are performed. These results will help to establish the relative importance that one should give to the diversification strategy versus the intensification strategy. Once again, if one observes the number of times each run is the overall best (Best ($*$)), then one may see that 6/1 is better on eight occasions, 3/2 on six, 2/3 on two and 1/6 on four. For a given level of effort (18 local branching subproblems), it would seem that by applying more diversification, one obtains better results. Therefore, in this case, the best strategy concerning the size of the meta phases is to set it equal to one.

N	$\overline{f}$	Category	nb. i.	2/1	4/1	6/1	8/1	L-shaped
60	1.025	*sol.*	1	1312.43 (248.25)	1311.71 (496.91)	1310.94 (873.40)	1310.54 (1150.11)	1310.06 (637.28)
		not sol.	0	-	-	-	-	-
		und.	0	-	-	-	-	-
	1.05	*sol.*	3	1343.63 (350.27)	1342.8 (757.1)	1342.24 (1128.43)	1340.85 (1520.49)	1340.91 (275.76)
		not sol.	0	-	-	-	-	-
		und.	0	-	-	-	-	-
	1.075	*sol.*	3	1360.17 (342.32)	1359.12 (730.58)	1357.93 (1093.31)	1357.73 (1491.66)	1358.67 (1488.22)
		not sol.	2	1293.21 (411.18)	1293.21 (820.96)	1293.21 (1231.25)	1293.21 (1641.56)	1281.72 (6034.74)
		und.	0	-	-	-	-	-
	1.10	*sol.*	2	1312.55 (395.91)	1311.68 (770.25)	1311.88 (1173.97)	1311.88 (1545.52)	1311.58 (3890.89)
		not sol.	3	1356.13 (386.24)	1352.73 (775.50)	1352.31 (1195.63)	1351.95 (1574.10)	1353.74 (6038.66)
		und.	0	-	-	-	-	-
70	1.025	*sol.*	3	1431.04 (388.04)	1430.79 (732.81)	1430.79 (1115.05)	1430.79 (1476.41)	1433.46 (157.34)
		not sol.	0	-	-	-	-	-
		und.	0	-	-	-	-	-
	1.05	*sol.*	2	1387.53 (395.17)	1385.72 (791.08)	1385.93 (1272.56)	1380.74 (1680.38)	1380.77 (369.70)
		not sol.	1	1427.08 (688.92)	1427.07 (1440.92)	1427.07 (2075.92)	1426.68 (2898.15)	1427.57 (6011.45)
		und.	0	-	-	-	-	-
	1.075	*sol.*	1	1401.03 (766.75)	1397.65 (1554.18)	1397.65 (2397.02)	1397.65 (3139.22)	1397.65 (2887.62)
		not sol.	4	1453.92 (698.36)	1451.52 (1408.84)	1451.05 (2152.15)	1450.27 (2886.54)	1455.58 (6020.53)
		und.	0	-	-	-	-	-
	1.10	*sol.*	0	-	-	-	-	-
		not sol.	4	1495.48 (790.11)	1493.25 (1568.15)	1492.70 (2328.58)	1492.90 (3178.22)	1493.06 (6011.10)
		und.	0	-	-	-	-	-
80	1.025	*sol.*	2	1481.78 (876.32)	1481.04 (1614.56)	1481.49 (2531.47)	1480.33 (3347.57)	1480.14 (106.97)
		not sol.	0	-	-	-	-	-
		und.	0	-	-	-	-	-
	1.05	*sol.*	1	1505.64 (1120.15)	1505.33 (2182.12)	1505.33 (3347.52)	1505.33 (4362.17)	1505.33 (4391.95)
		not sol.	1	1482.53 (1099.36)	1482.53 (2268.15)	1482.53 (3521.99)	1482.53 (4642.29)	1468.72 (6004.49)
		und.	0	-	-	-	-	-
	1.075	*sol.*	2	1392.86 (824.91)	1392.86 (1601.04)	1392.86 (2234.21)	1392.86 (3216.07)	1392.87 (41.24)
		not sol.	3	1549.60 (1098.36)	1547.70 (2262.29)	1544.37 (3387.57)	1544.09 (4625.32)	1554.97 (6027.58)
		und.	0	-	-	-	-	-
	1.10	*sol.*	2	1505.56 (939.47)	1505.47 (1980.21)	1502.50 (2994.12)	1500.63 (3984.65)	1497.24 (2656.43)
		not sol.	3	1483.51 (1144.09)	1481.45 (2340.40)	1481.51 (3512.41)	1480.00 (4765.84)	1497.28 (6012.34)
		und.	0	-	-	-	-	-
90	1.025	*sol.*	1	1638.93 (891.35)	1636.97 (1495.22)	1636.97 (2449.83)	1636.77 (3602.31)	1636.37 (1043.17)
		not sol.	1	1508.08 (770.21)	1508.08 (2005.50)	1507.34 (3174.54)	1507.04 (4283.28)	1510.94 (6084.15)
		und.	0	-	-	-	-	-
	1.05	*sol.*	2	1590.80 (1040.17)	1590.20 (2293.46)	1589.90 (3588.56)	1590.00 (4703.11)	1591.27 (2060.68)
		not sol.	2	1612.69 (1511.47)	1610.03 (2984.18)	1608.91 (4452.84)	1608.95 (6076.69)	1611.88 (6013.45)
		und.	1	1618.76 (1455.36)	1618.56 (3237.7)	1618.56 (5702.97)	1618.56 (7304.46)	- (6294.68)
	1.075	*sol.*	0	-	-	-	-	-
		not sol.	5	1601.41 (1571.91)	1600 (3174.45)	1597.96 (4757.07)	1596.95 (6377.13)	1601.17 (6007.51)
		und.	0	-	-	-	-	-
	1.10	*sol.*	0	-	-	-	-	-
		not sol.	5	1596.63 (1535.45)	1595.88 (3111.71)	1594.94 (4721.05)	1594.01 (6278.16)	1592.30 (6016.41)
		und.	0	-	-	-	-	-
Total		*sol.*	25	1538.67 (639.53)	1537.81 (1278.27)	1537.38 (1955.92)	1536.51 (2625.08)	1536.93 (1331.62)
		not sol.	34	1505.44 (1052.37)	1503.77 (2141.12)	1502.8 (3227.68)	1502.24 (4343.87)	1505.13 (6019.95)
		und.	1	1618.76 (1455.36)	1618.56 (3237.7)	1618.56 (5702.97)	1618.56 (7304.46)	- (6294.68)

Table 4: Results: Multi-descent algorithm vs. L-shaped

We will conclude this section by comparing the multi-descent heuristic with the L-shaped algorithm of Rei and *al.* [24]. The heuristic algorithm is applied by specifying the total number of meta phases of size one to be done. Again each descent will have a depth of three levels (m =3). As for the L-shaped algorithm, a maximum time of 6000 seconds is imposed for the solution process.

In Table 4 all instances solved are separated into three categories according to the results obtained by the L-shaped algorithm. The *sol.* category refers to all cases were the L-shaped algorithm was able to solve the problem for an optimality gap of $\epsilon \leq 1\%$. The *not sol.* category includes all instances that the L-shaped algorithm was unable to solve in the maximum time allowed but where at least one feasible solution was obtained. Finally, the *und.* category refers to the cases where the L-shaped algorithm was unable to solve the problem and no feasible solution was found before the maximum time allowed was reached. According to this classification, one obtains 25 instances in the *sol.* category, 34 in the *not sol.* category and only one in the *und.* category. In order to see how results vary for the multi-descent scheme, runs for the heuristic algorithm are made for two, four, six and eight meta phases. Results in Table 4 include both the best solution values found by both algorithms as well as the solution times in seconds, which are the values reported between parentheses.

If one considers those instances that are solved by the L-shaped algorithm (*sol.*), one first observes that the solutions obtained by the multi-descent heuristic are in almost all cases either optimal or near optimal. For this category, the total mean results show that the 2/1 runs obtain an average value of 1538.67 in 639.53 seconds of computation time. As for the L-shaped algorithm, it obtains 1536.93 in 1331.62 seconds. The heuristic finds near optimal solutions in half the computation time when compared to the exact algorithm. Furthermore, as the number of meta phases increases, the quality of results converge to the optimal values. The 8/1 runs produce a total average value of 1536.51 versus 1536.93 for the L-shaped algorithm, whose results have an average gap of less than or equal to one percent. One should note that the total average computation time of 8/1 is larger than that of the exact algorithm (2625.08 seconds versus 1331.62 seconds). However, what these results seem to indicate is that the heuristic is robust since it generates near optimal solutions relatively quickly, and in any case, if the number of meta phases is increased then the procedure converges to optimality.

We will now analyse the results obtained on those instances that were not solved by the L-shaped algorithm (*not sol.* and *und.*). The detailed results show that the multi-descent heuristic obtains better results in ten cases compared to only three for the exact algorithm. If one considers those instances for which $\overline{f} =$ 1.075 and 1.10, then the heuristic is usually better than the L-shaped algorithm and the differences can be quite significant. The multi-descent algorithm can obtain similar results a lot faster, as is the case for the three instances of size $N = 60$ and $\overline{f} = 1.10$ where the average results are 1351.95 in 1574.10 seconds for 8/1 compared to 1353.74 in 6038.66 seconds for the L-shaped algorithm; or, it can obtain better results in favorable times, as in the case of the three instances of size $N = 80$ and $\overline{f} = 1.10$ where the average results are 1480.00 in 4765 seconds for 8/1 compared to 1497.28 in 6012.34 seconds for the exact algorithm. Out of the three cases where the exact algorithm obtained better results, only two are important to analyse ($N = 60$, $\overline{f} = 1.075$ and $N = 80$, $\overline{f}$

= 1.05). In the case of $N = 90$ and $\overline{f} = 1.10$, results are quite similar, 1594.01 in 6278.16 seconds for 8/1 versus 1592.30 in 6016.41 seconds for the L-shaped algorithm. For the two instances for which $N = 60$ and $\overline{f} = 1.075$, the 8/1 runs obtain 1293.21 in 1641.56 seconds compared to 1281.72 in 6034.74 seconds for the L-shaped algorithm. In this case, the computation times are not comparable. If one increases the number of meta phases to 16/1, then the results obtained are 1282.03 in 3355.66 seconds. Once again, the heuristic produces comparable results in a favorable time. For the single instance of size $N = 80$ and $\overline{f} = 1.05$, the same observation could not be made. In this case, the exact algorithm obtains 1468.72 in 6004.49 seconds. The 8/1 runs produce 1482.53 in 4642.29 seconds and when the number of meta phases is increased to 16/1, the results are 1478.56 in 9667.59 seconds. For this problem, the strategy used in the multi-descent scheme was unable to outperform the L-shaped algorithm. However, it remains a single case out of the 60 instances tested. The total mean results show that the multi-descent algorithm produces equivalent results in a lot less time, the 2/1 runs obtain 1505.44 in 1052.37 seconds compared to 1505.13 in 6019.95 seconds for the L-shaped algorithm. As one increases the number of meta phases, the results obtained are improved while the computation times remain favorable, the 8/1 runs obtain 1502.24 in 4343.87 seconds. Finally, the single instance in the *und* category can be used to illustrate that the heuristic is able to produce solutions quickly even if the exact algorithm was unable to.

6 Conclusion

In this paper, we propose a new hybrid algorithm that combines both local branching and Monte Carlo sampling in a multi-descent search strategy for integer 0-1 stochastic programming problems. By controling simultaneously the inherent complexities associated with both the first stage problem and the recourse function, one is able to better limit the effort needed to solve the approximating subproblems. Furthermore, by using the local branching constraints in order to obtain diversification for the search strategy, one is able to better explore the feasible region of the original problem. This method was specialized to the case of the SVRPSD and was proven to be quite effective to solve hard instances of the problem. One should also point out that the algorithmic principles that were used are all quite general. In future work, it would be interesting to see how one could adapt these ideas to other stochastic programming problems.

References

[1] S. Ahmed and A. Shapiro. The sample average approximation method for stochastic programs with integer recourse. Optimization Online,

http://www.optimization-online.org, February 2002.

[2] L. Bianchi, 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. Technical report, IDSIA / USI-SUPSI Dalle Molle Institute for Artificial Intelligence, Galleria 2, 6928 Manno, Switzerland, March 2005.

[3] J.R. Birge and F. Louveaux. *Introduction to Stochastic Programming.* Springer, 1997.

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

[5] G.B. Dantzig and P.W. Glynn. Parallel processors for planning under uncertainty. *Annals of Operations Research*, 22:1–21, 1990.

[6] M. Dror. Modeling vehicle routing with uncertain demands as a stochastic program: Properties of the corresponding solution. *European Journal of Operational Research*, 64:432–441, 1993.

[7] M. Dror, G. Laporte, and P. Trudeau. Vehicle routing with stochastic demands: properties and solution frameworks. *Transportation Science*, 23(3):166–176, 1989.

[8] Y. Ermoliev. Stochastic quasigradient methods. In *Numerical techniques for stochastic optimization*, pages 141–186. Springer-Verlag, Berlin, 1988.

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

[10] M.C. Fu. Optimization via simulation: a review. *Annals of Operations Research*, 53:199–247, 1994.

[11] M.C. Fu. Optimization for simulation theory vs. practice. *INFORMS Journal on Computing*, 14(3):192–215, 2002.

[12] M. Gendreau, G. Laporte, and R. Séguin. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science*, 29(2):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(3), 1996.

[14] G. Gutin and A.P. Punnen, editors. *The Traveling Salesman Problem and Its Variations*. Springer, 2002.

[15] J.L. Higle and S. Sen. *Stochastic Decomposition A statistical method for large scale stochastic linear programming.* Kluwer Academic Publishers, 1996.

[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] A.J. Kleywegt, A. Shapiro, and T. Hommem-De-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2001.

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

[19] G. Laporte, F.V. Louveaux, and L. Van Hamme. An integer l-shape algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50(3):415–423, 2002.

[20] J. Linderoth, A. Shapiro, and S. Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142:215–241, 2006.

[21] F. Louveaux. An introduction to stochastic transportation models. In M. Labbe, G. Laporte, K. Tanczos, and P. Toint, editors, *Operations Research and Decision Aid Methodologies in Traffic and Transportation Management*, volume 166 of *Computer and Systems Sciences*, pages 244–263. Springer-Verlag, 1998.

[22] J. Lysgaard, A.N. Letchford, and R.W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100:423–445, 2004.

[23] W-K. Mak, D. Morton, and R.K. Wood. Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, pages 47–56, 1999.

[24] W. Rei, M. Gendreau, and P. Soriano. Local branching cuts for the 0-1 integer l-shaped algorithm. Technical report, Center for Research on Transportation, C.P. 6128, succursale Centre-ville Montréal QC H3C 3J7 Canada, September 2006.

[25] T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167:96–115, 2005.

[26] N. Secomandi. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers & Operations Research*, 27:1201–1225, 2000.

[27] N. Secomandi. A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research*, 49(5):796–802, 2001.

[28] A. Shapiro and T. Hommem-De-Mello. On the rate of convergence of optimal solutions of monte carlo approximations of stochastic programs. *SIAM Journal on Optimization*, 11(1):70–86, 2000.

[29] R.M. Van Slyke and R. Wets. L-shaped programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663, 1969.

[30] P. Toth and D. Vigo, editors. *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. SIAM, 2001.

[31] B. Verweij, S. Ahmed, A.J. Kleywegt, G. Nemhauser, and A. Shapiro. The sample average approximation method applied to stochastic routing problems: a computational study. *Computational Optimization and Applications*, 24:289–333, 2003.

[32] W. Yang, K. Mathur, and R.H. Ballou. Stochastic vehicle routing problem with restocking. *Transportation Science*, 34(1):99–112, 2000.