CIRRELT

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

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

# Local Branching Cuts for the 0-1 Integer L-Shaped Algorithm

**Walter Rei**
**Michel Gendreau**
**Patrick Soriano**

**July 2007**

**CIRRELT-2007-23**

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

**Bureaux de Québec :**
Université Laval
Pavillon Palasis-Prince, local 2642
Québec (Québec)
Canada G1K 7P4
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

# Local Branching Cuts for the 0-1 Integer L-Shaped Algorithm

**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.** Local branching has been presented as a new solution strategy for hard to solve mixed integer problems. Research has recently been done on using local branching as a way to generate optimality cuts in the case of Benders decomposition. In this paper, local branching is used to produce a new type of valid inequalities for the case of the 0-1 integer L-shaped algorithm. Numerical tests were conducted on a series of stochastic routing problems. Results prove the usefulness of these new inequalities.

**Keywords**. Stochastic programming, local branching, valid inequalities, vehicle routing problems with stochastic demands.

**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 #1301 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

Integer stochastic problems with fixed recourse can be defined as follows:

$$\text{Min} \quad c^\top x + \mathcal{Q}(x) \tag{1}$$
$$\text{s.t.} \quad Ax = b \tag{2}$$
$$x \in X, \tag{3}$$
$$\text{where } \mathcal{Q}(x) = \mathbf{E}_\xi[Q(x, \xi(\omega))] \tag{4}$$
$$\text{and } Q(x, \xi(\omega)) = Min_y\{q(\omega)^\top y \mid Wy = h(\omega) - T(\omega)x,\ y \in Y\}. \tag{5}$$

Problem (1)-(5) has two decision stages. Vector $x$ represents the first stage. These decisions must be made before a series of stochastic parameters, represented by $\xi$, become known. In the second stage, one observes a certain random event, $\omega \in \Omega$, and $\xi$ is fixed to $\xi(\omega)$. At this point, parameters $q(\omega)$, $h(\omega)$ and $T(\omega)$ that make up $\xi(\omega)$, become known. Problem (1)-(5) is said to have fixed recourse because matrix $W$ is not dependent on the random event. In the second stage, vector $y$ represents the recourse that one can take given that the first stage decisions were fixed at $x$. Function $Q : x \times \xi(\omega) \to \mathbb{R}_+$ measures the objective value of the recourse decisions. Therefore, function $\mathcal{Q} : x \to \mathbb{R}_+$, called the recourse function, gives the average value of recourse for $x$ over all $\omega \in \Omega$. Problem (1)-(5) consists of finding a solution $x$ that is minimal, on average, when considering $\xi$. Integrality constraints may be present in sets $X$ and $Y$. As for the probability distribution of $\xi$, it may be either discrete or continuous.

Integer stochastic models have been successfully applied to a variety of contexts [4]. The algorithms that have been used to solve problems of type (1)-(5) have either been based on cutting plane strategies ([23], [2], [13], [6], [7], [21], [19] and [20]), branch-and-bound procedures ([16], [5] and [1]) or more recently on the use of test sets ([18] and [11]).

In this paper, we propose to use local branching as a way to generate a new type of cuts that can be introduced in the case of the 0-1 integer L-shaped method. This follows from the research initiated in Rei and *al.* [17] for the case of general Benders decomposition that showed how local branching could be used to simultaneously improve the upper and lower bounds in the Benders algorithm. Similar ideas can be applied to the relaxation of the recourse function $\mathcal{Q}(x)$ in the case of the L-shaped branch-and-cut algorithm. The principles evoked here are applicable to all problems of type (1)-(5) having a subset of binary $x$ variables. In this paper, however, they are illustrated and validated in the context of the single vehicle routing problem with stochastic demands (see [9], [12] and [14]), for which they lead to a significantly more efficient algorithm. The effectiveness is clearly demonstrated by computational experiments performed on a large set of instances whose difficulty ranges from easy to extremely difficult.

The remainder of this article is organized as follows. In section 2, a description of the 0-1 integer L-shaped method is given. Section 3 follows by providing an explanation of how a local branching search strategy may be applied to the case of the 0-1 integer L-shaped algorithm as well as a presentation of the valid inequalities that one can derive from it. In section 4, a presentation of the one vehicle routing problem is given as well as a description of the various cuts that can be applied to this problem. This is followed by computational results in section 5. Finally, this article is concluded in section 6.

## 2 The 0-1 integer L-shaped algorithm

The L-shaped algorithm was first introduced in 1969 by Van Slyke and Wets [22] to solve continuous stochastic programming problems with fixed recourse, in which random events are represented by a finite set of discrete scenarios. This algorithm can be interpreted as the application of Benders decomposition to the extensive form of the stochastic problem (see [4]). Indeed, when integrality constraints are introduced in set $X$, (1)-(5) becomes a mixed integer program and the L-shaped algorithm is then equivalent to the original Benders decomposition algorithm [3]. The problem becomes significantly harder when integrality constraints are introduced in set $Y$ (integer recourse). In the Benders decomposition algorithm, cuts are created using the dual information provided by the solution of the subproblem. In the stochastic case, this subproblem is defined as follows: $Min_y\{q(\omega)^\top y \ \mid \ Wy = h(\omega) - T(\omega)x, \ y \in Y\}$ for all random events. One must realize that when integrality constraints are present in $Y$, one can no longer apply standard continuous duality. Caroe and Tind [7] show how the L-shaped algorithm can be generalized to the case of integer recourse by using general duality theory. Convergence is maintained when either cutting plane algorithms or branch-and-bound procedures are used to solve the stochastic subproblems.

The 0-1 integer L-shaped method, which tackles problems with binary first stage variables and integer recourse, was introduced by Laporte and Louveaux [13]. In the following, we use the notation defined in their paper. Let us assume that vector $x$ is of size $n_1$ and that set $X$ is defined as follows: $X = \overline{X} \cap \{0,1\}^{n_1}$. Then let the following problem be the *current problem*:

$$\text{Min} \quad c^\top x + \Theta \tag{6}$$

$$\text{s.t.} \quad Ax = b \tag{7}$$

$$D_k x \geq d_k, \ k = 1, \ldots, s, \tag{8}$$

$$E_l x + \Theta \geq e_l, \ l = 1, \ldots, t, \tag{9}$$

$$0 \leq x \leq 1, \ \Theta \in \mathbb{R}. \tag{10}$$

Constraints of type (8) are called feasibility cuts, since they are used to induce feasible values of $x$. The authors define a set of feasibility cuts to be valid if

there is a finite value $s$ such that $x \in \overline{X}$ if and only if $\{D_k x \geq d_k,\ k = 1, \ldots, s\}$. Constraints of type (9) are said to be optimality cuts since they express possible feasible values of the recourse function. As before, a set of $t$ optimality cuts is valid if the following condition is verified: $\forall x \in X,\ (x, \Theta) \in \{(x, \Theta) \mid E_l x + \Theta \geq e_l,\ l = 1, \ldots, t\}$ implies $\Theta \geq \mathcal{Q}(x)$.

One can now define the 0-1 integer L-shaped algorithm as follows:

***0-1 Integer L-shaped Algorithm***

**Step 0 (Initialization)**
Set $\nu = 0$, $t = 0$, $s = 0$,
$\overline{z} = +\infty$,
$\Theta = -\infty$ or any valid general lower bound $L$.
Define the first pendant node as the initial current problem.

**Step 1 (Selection)**
Using a selection criterion, select a pendant node,
**if** there is none **STOP**.

**Step 2 (Separation)**
**(2.1)** $\nu = \nu + 1$.
**(2.2)** Solve the current problem,
**if** the current problem is infeasible **then**
  fathom the node and go to **Step 1**,
**else**
  let $(x^\nu, \Theta^\nu)$ be the optimal solution to the problem.
**end if**

**(2.3)** Search for violated constraints of type (8),
**if** one is found **then**
  add it to the current problem, set $s = s + 1$ go to **(2.2)**,
**else if** $c^\top x^\nu + \Theta^\nu > \overline{z}$ **then**
  fathom the node and go to **Step 1**.
**end if**

**(2.4)** Search for violated integrality constraints,
**if** one is found **then**
  go to **Step 3**,
**else**
  solution $x^\nu$ is feasible.
**end if**

**(2.5)** Compute $\mathcal{Q}(x^\nu)$, $z^\nu = c^\top x^\nu + \mathcal{Q}(x^\nu)$, $\overline{z} = Min\{\overline{z}, z^\nu\}$,
**if** $\Theta^\nu \geq \mathcal{Q}(x^\nu)$ **then**
  fathom the node and go to **Step 1**,
**else**

add an optimality cut (9), set $t = t + 1$ go to **(2.2)**.
**end if**

**Step 3 (Branching)**
Using a branching criterion, create two new nodes, append them to the list of active nodes and go to **Step 1**.

The 0-1 integer L-shaped algorithm converges in a finite number of steps whenever a valid set of optimality and feasibility cuts exists. A general lower bound $\underline{z}$ can be obtained, at any point of the solution process, by considering the minimum of the lower bounds associated with the currently active nodes. This allows stopping the search procedure when an $\epsilon - opt$ solution is found.

Optimality cuts of type (9) can be expressed as follows: let us first define the $r - th$ feasible solution generated by the method as being $x_i = 1$, $i \in S_r$ and $x_i = 0$, $i \notin S_r$ and let $\Theta_r$ be the recourse value associated with this feasible solution. Laporte and Louveaux [13] show that the set of cuts defined for all feasible solutions $r$:

$$\Theta \geq (\Theta_r - L)\Big(\sum_{i \in S_r} x_i - \sum_{i \notin S_r} x_i\Big) - (\Theta_r - L)(|S_r| - 1) + L \qquad (11)$$

where value $L$ is a general lower bound on the value of recourse, is a valid set of optimality cuts. The proof of validity is evident when one considers that $\sum_{i \in S_r} x_i - \sum_{i \notin S_r} x_i$ is always less than or equal to $|S_r|$. The information provided by cut (11) is that whenever the solution considered is the $r-th$ feasible solution then the recourse is equal to $\Theta_r$, else the value of recourse is less than $L$. One can also express the optimality cuts in the following way:

$$\sum_{i \in S_r} x_i \leq |S_r| - 1. \qquad (12)$$

In the form of equation (12), the optimality cut is only used to eliminate from further consideration the $r - th$ feasible solution. In this case it provides no information on the value of recourse. However, it has the advantage of being composed of coefficients equal to one, which makes cut (12) more numerically stable than (11). As for the feasibility cuts of type (8), one can only say that they are problem dependent.

The main difficulty associated with the 0-1 integer L-shaped algorithm resides in the approximation of the recourse function. The information provided by optimality cuts of either type (11) or (12) is very local. Constraints (11) only bound the value of recourse associated with the feasible solutions that were used to create them. As for (12), they only eliminate the feasible solutions that were encountered by the algorithm. Therefore, in problem (6)-(10), the value of recourse will only be determined by the subset of optimality constraints present and the value of the general bound $L$. If one uses exclusively (12), then only the

general bound $L$ approximates the recourse function. If the quality of bound $L$ is poor, then the L-shaped algorithm will tend to enumerate feasible solutions. The reason for this is that in the separation step (more precisely in **(2.3)**), the quality of bound $c^\top x^\nu + \Theta^\nu$ will be poor whenever $x^\nu$ is infeasible, resulting in an increase of the number of active subproblems. To solve this problem, one can either provide better general lower bounds $L$ or use other lower bounding functionals to better approximate the recourse function. In [13], an improved optimality cut is presented where if a better lower bound exists in a close neighbourhood of the feasible solution considered, then (11) is strenghtened by this information. In the next section, a series of general valid inequalities will be presented to extend this idea of using better lower bounds around solutions to the *current problem*. These valid inequalities will help to provide a better description of the recourse values for different neighbourhoods of the first stage feasible region.

# 3 Local branching cuts for the 0-1 integer L-shaped algorithm

Local branching was introduced in 2003 by Fischetti and Lodi [8]. 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 efficiently each of the subregions thus created.

In the case of 0-1 integer problems, the function used to divide the feasible region is the Hamming distance defined from a particular integer point. Let us consider the general stochastic problem (1)-(5), for which $X = \overline{X} \cap \{0,1\}^{n_1}$. Let $x^0$ be a vector of 0-1 values, then the Hamming distance function relative to $x^0$ is: $\Delta(x, x^0) = \sum_{j \in S_0} (1 - x_j) + \sum_{j \in N_1 \setminus S_0} x_j$ (where $N_1 = \{1, \ldots, n_1\}$ and $S_0 = \{j \in N_1 \mid x^0_j = 1\}$). Using function $\Delta(x, x^0)$, for any integer $\kappa$, one can divide the feasible region of (1)-(5) by creating two subproblems, one in which the constraint $\Delta(x, x^0) \leq \kappa$ is added, and the other in which $\Delta(x, x^0) \geq \kappa + 1$ is added. Constraint $\Delta(x, x^0) \leq \kappa$ can considerably reduce the size of the feasible region of problem (1)-(5) when $\kappa$ is fixed to an appropriate (small) value. Therefore, one can use an adapted generic solver to solve this subproblem. Using the new solution found, the procedure may continue by dividing the subregion defined by $\Delta(x, x^0) \geq \kappa + 1$ into two more subproblems where the smaller subregion is explored in the same way as before.

In [17], the information associated with a local branching descent is used to produce multiple optimality cuts in the case of classical Benders decomposition. Through this process, one is also able to improve the upper bound generated by

the Benders algorithm. Furthermore, it is shown how local branching constraints can be used to replace or complement the feasibility cuts. We will use similar ideas to generate a series of valid inequalities that approximate the value of recourse in different subregions of $X$. Let us first define the two following subproblems:

$$(P_n) \quad \begin{array}{ll} \text{Min} & c^\top x + \mathcal{Q}(x) \\ \text{s.t} & Ax = b \\ & \Delta(x, x^i) \geq \kappa_i,\ i \in I^n \\ & \Delta(x, x^n) \leq \kappa_n \\ & x \in X \end{array} \qquad (\overline{P}_n) \quad \begin{array}{ll} \text{Min} & c^\top x + \mathcal{Q}(x) \\ \text{s.t} & Ax = b \\ & \Delta(x, x^i) \geq \kappa_i,\ i \in I^n \\ & \Delta(x, x^n) \geq \kappa_n + 1 \\ & x \in X, \end{array}$$

where $I^n$ corresponds to a set of 0-1 vectors that may or may not represent feasible first stage solutions. A local branching step consists of solving subproblem $P_n$ and then using the solution obtained to separate the feasible region of subproblem $\overline{P}_n$. If $P_n$ is found to be infeasible, a diversification strategy is applied and the problem is solved again. In this case, the diversification strategy consists of increasing value $\kappa_n$ to obtain a larger feasible region for $P_n$.

A local branching descent is composed of a series of subproblems $P_n$ for $n = 1, \ldots, m$ that correspond to the subregions explored by the algorithm. Let us suppose that one is able to find a lower bound $\overline{\Theta}_n$ (for which $\overline{\Theta}_n > L$) on the value of the recourse in each of the subregions associated with subproblems $P_n$, $n = 1, \ldots, m$. Let us also redefine the *current problem* in the following way:

$$\text{Min} \quad c^\top x + \Theta \tag{13}$$

$$\text{s.t.} \quad Ax = b \tag{14}$$

$$D_k x \geq d_k,\ k = 1, \ldots, s, \tag{15}$$

$$E_l x + \Theta \geq e_l,\ l = 1, \ldots, t, \tag{16}$$

$$\Delta(x, x^i) \geq 1,\ i \in \bigcup_{n=1}^{m} I^n, \tag{17}$$

$$0 \leq x \leq 1,\ \Theta \in \mathbb{R}. \tag{18}$$

In (17), one should realize that constraint $\Delta(x, x^i) \geq 1$ eliminates locally vector $x^i$ since it imposes that the Hamming distance between $x$ and $x^i$ be at least one. If $x^i$ is feasible, then $\Delta(x, x^i) \geq 1$ is equivalent to the optimality cut (12). If $x^i$ is infeasible, then $\Delta(x, x^i) \geq 1$ may serve as a feasibility cut. Therefore, model (13)-(18) is either equivalent to (6)-(10) or it offers a better description of the feasible region. One can now derive the following result:

**Proposition 3.1** (Local branching valid inequalities)**.** *Let $P_n$, $n = 1, \ldots, m$, define a local branching descent and $\overline{\Theta}_n$, $n = 1, \ldots, m$, be valid lower bounds on the recourse value for each of the subproblems in the descent, then the following*

*system of equations defines a set of valid inequalities for problem (13)-(18):*

$$\Theta \geq L + (\overline{\Theta}_n - L)w_n,\ n = 1, \ldots, m \tag{19}$$

$$\kappa_n - \Delta(x, x^n) \leq n_1 \sum_{j=1}^{n} w_j,\ n = 1, \ldots, m \tag{20}$$

$$\Delta(x, x^n) - \kappa_n \leq (1 - w_n)n_1,\ n = 1, \ldots, m \tag{21}$$

$$w_{n+1} \leq \overline{w}_n,\ n = 1, \ldots, m-1 \tag{22}$$

$$\sum_{j=1}^{n-1} w_j + \overline{w}_{n-1} = 1,\ n = 2, \ldots, m \tag{23}$$

$$w_n \in \{0, 1\},\ n = 1, \ldots, m \tag{24}$$

$$\overline{w}_n \in \{0, 1\}, n = 1, \ldots, m-1, \tag{25}$$

*where $x^n$, $n = 1, \ldots, m$, is a series of 0-1 vectors (that may or may not be feasible first stage solutions), such that $x^n \in \{x \mid \Delta(x, x^{n-1}) \leq \kappa_{n-1}, \Delta(x, x^i) \geq \kappa_i + 1, i \in I^{n-1}\}$, $n = 2, \ldots, m$.*

*Proof.* The proposition will be proved by induction.

- Base case: (19)-(25) is a set of valid inequalities for (13)-(18) when $m = 1$.

If $m = 1$, then system (19)-(25) reduces to:

$$\Theta \geq L + (\overline{\Theta}_1 - L)w_1, \tag{26}$$

$$\kappa_1 - \Delta(x, x^1) \leq w_1 n_1, \tag{27}$$

$$\Delta(x, x^1) - \kappa_1 \leq (1 - w_1)n_1, \tag{28}$$

$$w_1 \in \{0, 1\}. \tag{29}$$

Let $\hat{x}$ be a feasible solution to problem (13)-(18). One should note that $\hat{x}$ is either integer or continuous and either feasible or not with respect to the original problem (1)-(5). There are three possible cases when one considers $\hat{x}$:

1. $0 < \Delta(\hat{x}, x^1) < \kappa_1$,
2. $0 < \kappa_1 < \Delta(\hat{x}, x^1)$,
3. $\Delta(\hat{x}, x^1) = \kappa_1$.

Case 1: if $0 < \Delta(\hat{x}, x^1) \leq \kappa_1$, then constraint (27) forces $w_1 = 1$. If $w_1 = 1$, then (26) becomes $\Theta \geq \overline{\Theta}_1$ which is valid since $\Delta(\hat{x}, x^1) \leq \kappa_1$. Also, in this case, if $w_1 = 1$, then constraint (28) is satisfied.

Case 2: if $0 < \kappa_1 < \Delta(\hat{x}, x^1)$, then constraint (28) forces $w_1 = 0$. If $w_1 = 0$, then (26) becomes $\Theta \geq L$ which is valid. Also, in this case, if $w_1 = 0$ then

constraint (27) is satisfied.

Case 3: if $\Delta(\hat{x}, x^1) = \kappa_1$, then constraints (27) and (28) allow $w_1 = 1$ or $w_1 = 0$. If $w_1 = 1$, then (26) becomes $\Theta \geq \overline{\Theta}_1$ which is valid since $\Delta(\hat{x}, x^1) \leq \kappa_1$. Otherwise, if $w_1 = 0$, then (26) becomes $\Theta \geq L$ which is also valid. Considering that the *current problem* is a minimization problem and since $\overline{\Theta}_1 > L$, then, in this case, variable $w_1$ is fixed to 0 ($w_1 = 0$) and (26) becomes $\Theta \geq L$.

- Hypothesis $H1$: (19)-(25) is a set of valid inequalities for (13)-(18) when $m = i$.
- Step: if $H1$ is true then (19)-(25) is a set of valid inequalities for (13)-(18) when $m = i + 1$.

If $m = i$, then system (19)-(25) becomes:

$$\Theta \geq L + (\overline{\Theta}_n - L)w_n,\ n = 1, \ldots, i \tag{30}$$

$$\kappa_n - \Delta(x, x^n) \leq n_1 \sum_{j=1}^{n} w_j,\ n = 1, \ldots, i \tag{31}$$

$$\Delta(x, x^n) - \kappa_n \leq (1 - w_n)n_1,\ n = 1, \ldots, i \tag{32}$$

$$w_{n+1} \leq \overline{w}_n,\ n = 1, \ldots, i - 1 \tag{33}$$

$$\sum_{j=1}^{n-1} w_j + \overline{w}_{n-1} = 1,\ n = 2, \ldots, i \tag{34}$$

$$w_n \in \{0, 1\},\ n = 1, \ldots, i \tag{35}$$

$$\overline{w}_n \in \{0, 1\}, n = 1, \ldots, i - 1. \tag{36}$$

If one adds another level to the local branching descent (from $m = i$ to $m = i + 1$), one adds to (30)-(36) the following inequalities:

$$\Theta \geq L + (\overline{\Theta}_{i+1} - L)w_{i+1}, \tag{37}$$

$$\kappa_{i+1} - \Delta(x, x^{i+1}) \leq n_1 \sum_{j=1}^{i+1} w_j, \tag{38}$$

$$\Delta(x, x^{i+1}) - \kappa_{i+1} \leq (1 - w_{i+1})n_1, \tag{39}$$

$$w_{i+1} \leq \overline{w}_i, \tag{40}$$

$$\sum_{j=1}^{i} w_j + \overline{w}_i = 1, \tag{41}$$

$$w_{i+1} \in \{0, 1\}, \tag{42}$$

$$\overline{w}_i \in \{0, 1\}. \tag{43}$$

Let us now define the following set: $X_i = \bigcup_{n=1}^{i} \{x \mid \Delta(x, x^n) \leq \kappa_n\}$. Once more, let $\hat{x}$ be a feasible solution to problem (13)-(18). There are two possible cases when one considers $\hat{x}$:

1. $\hat{x} \notin X_i$,
2. $\hat{x} \in X_i$.

Case 1: if $\hat{x} \notin X_i$ (i.e. $\Delta(\hat{x}, x^n) > \kappa_n$ for $n = 1, \ldots, i$), then (32) forces $w_n = 0$, $n = 1, \ldots, i$. When one considers system (37)-(43), one must first see that since $w_n = 0$, $n = 1, \ldots, i$, then (41) implies that $\overline{w}_i = 1$. Constraint (40) becomes $w_{i+1} \leq 1$ which makes it possible for variable $w_{i+1}$ to be equal to either 0 or 1. Since $\hat{x} \notin X_i$, then three cases are possible:

- $0 < \Delta(\hat{x}, x^{i+1}) < \kappa_{i+1}$,
- $0 < \kappa_{i+1} < \Delta(\hat{x}, x^{i+1})$,
- $\Delta(\hat{x}, x^{i+1}) = \kappa_{i+1}$.

Since $w_n = 0$, $n = 1, \ldots, i$, if $0 < \Delta(\hat{x}, x^{i+1}) < \kappa_{i+1}$, then constraint (38) forces $w_{i+1} = 1$. If $w_{i+1} = 1$, then (37) becomes $\Theta \geq \overline{\Theta}_{i+1}$ which is valid since $\Delta(\hat{x}, x^{i+1}) \leq \kappa_{i+1}$. Also, if $w_{i+1} = 1$, then constraint (39) is satisfied. If $0 < \kappa_{i+1} < \Delta(\hat{x}, x^{i+1})$, then constraint (39) forces $w_{i+1} = 0$. If $w_{i+1} = 0$, then (37) becomes $\Theta \geq L$ which is valid. Also, in this case, if $w_{i+1} = 0$ then constraint (38) is satisfied. If $\Delta(\hat{x}, x^{i+1}) = \kappa_{i+1}$, then constraints (38) and (39) allow $w_{i+1} = 1$ or $w_{i+1} = 0$. If $w_{i+1} = 1$, then (37) becomes $\Theta \geq \overline{\Theta}_{i+1}$ which is valid since $\Delta(\hat{x}, x^{i+1}) \leq \kappa_{i+1}$. Otherwise, if $w_{i+1} = 0$, then (37) becomes $\Theta \geq L$ which is also valid. Once again, considering that the *current problem* is a minimization problem and since $\overline{\Theta}_{i+1} > L$, then, in this case, variable $w_{i+1}$ is fixed to 0 ($w_{i+1} = 0$) and (37) becomes $\Theta \geq L$.

Case 2: if $\hat{x} \in X_i$, then there are two possible cases to consider:

1. there exists at least one index $n$, $1 \leq n \leq i$, such that $\Delta(\hat{x}, x^n) < \kappa_n$,
2. $\Delta(\hat{x}, x^n) \not< \kappa_n$, for $n = 1, \ldots, i$.

Subcase 1: If there exists at least one index $n$, $1 \leq n \leq i$, such that $\Delta(\hat{x}, x^n) < \kappa_n$, then constraints (31) and (34) imply that there exists exactly one index $1 \leq i^\star \leq i$ such that $w_{i^\star} = 1$. If $w_{i^\star} = 1$, then (30) reduces to $\Theta \geq \overline{\Theta}_{i^\star}$, which is valid because of $H1$. Considering system (37)-(43), since $w_{i^\star} = 1$, constraint (41) implies that $\overline{w}_i = 0$. Then by (40), $w_{i+1} = 0$. If $w_{i+1} = 0$, then (37) reduces to $\Theta \geq L$ which is valid. One must now consider two possible cases:

- $0 < \Delta(\hat{x}, x^{i+1}) \leq \kappa_{i+1}$,
- $0 < \kappa_{i+1} < \Delta(\hat{x}, x^{i+1})$.

If $0 < \Delta(\hat{x}, x^{i+1}) \leq \kappa_{i+1}$, then (38) is satisfied since $i^\star < i + 1$. If $0 < \kappa_{i+1} < \Delta(\hat{x}, x^{i+1})$, since $w_{i+1} = 0$, then (39) is satisfied.

Subcase 2: if $\Delta(\hat{x}, x^n) \not< \kappa_n$, for $n = 1, \ldots, i$, and $\hat{x} \in X_i$, then, there exists an index set $J \subseteq \{1, \ldots, i\}$ such that $\Delta(\hat{x}, x^j) = \kappa_j$, $\forall j \in J$, and $\Delta(\hat{x}, x^j) > \kappa_j$, $\forall j \in \{1, \ldots, i\} \setminus J$. Without loss of generality, let us assume that $\Delta(\hat{x}, x^n) = \kappa_n$

for $n = 1, \ldots, j$, and $\Delta(\hat{x}, x^n) > \kappa_n$ for $n = j+1, \ldots, i$. Constraints (32) imply that $w_n = 0$ for $n = j+1, \ldots, i$. As for $n = 1, \ldots, j$, constraints (31) and (32) are always satisfied. Constraints (34) and (33) imply two possible alternatives: either there exists exactly one index $1 \leq i^\star \leq j$ such that $w_{i^\star} = 1$ and hence $\overline{w}_n = 1$ for $n = 1, \ldots, i^\star - 1$ **or else** $\overline{w}_n = 1$ for $n = 1, \ldots, j, \ldots, i-1$ and $w_n = 0$ for $n = 1, \ldots, j, \ldots, i$. Let us consider the first alternative, if $w_{i^\star} = 1$ and $\overline{w}_n = 1$ for $n = 1, \ldots, i^\star - 1$ then (30) reduces to $\Theta \geq \overline{\Theta}_{i^\star}$. Otherwise, if $\overline{w}_n = 1$ for $n = 1, \ldots, i-1$ and $w_n = 0$ for $n = 1, \ldots, i$ then (30) reduces to $\Theta \geq L$. Once again, since the *current problem* is a minimization problem and given that $\overline{\Theta}_n > L$ for $n = 1, \ldots, i$ (hence $\overline{\Theta}_{i^\star} > L$), therefore the second alternative will always be selected and (30) becomes $\Theta \geq L$, which is valid. This situation is identical to the one encountered in case 1 (i.e., case where $x \notin X_i$) and one must therefore consider the same three possibilities:

- $0 < \Delta(\hat{x}, x^{i+1}) < \kappa_{i+1}$,
- $0 < \kappa_{i+1} < \Delta(\hat{x}, x^{i+1})$,
- $\Delta(\hat{x}, x^{i+1}) = \kappa_{i+1}$.

For each of these, the validity of system (37)-(43) can be proven as before and will not be repeated here. □

The system of inequalities (19)-(25) is used to bound the value of recourse following the local branching descent. If one considers a feasible solution $\hat{x}$ to problem (13)-(18), then a valid lower bound on the value of recourse is provided for $\hat{x}$ in the first subproblem $P_n$, $n = 1, \ldots, m$, for which solution $\hat{x}$ is feasible. Otherwise, the value of recourse is bounded by $L$. If $\overline{\Theta}_n > L$ for $n = 1, \ldots, m$, then (19)-(25) offers a better description of the possible values of the recourse function in the subregions explored in the local branching descent. Furthermore, since (19)-(25) bounds $\mathcal{Q}(x)$ in different subregions of $X$ compared to only bounding the recourse associated with a feasible solution as in the case of the classical optimality cut (11), by using (19)-(25), one may obtain useful information on a wider range of solutions for problem (13)-(18).

One must now find a way to generate lower bounds on the subproblems explored during the local branching descent. To do so, one can either use a lower bounding functional that is specific to the problem being solved or, one can adopt a more general approach. It is observed in [17] that the local branching subproblems retain the same structure as the original problem. Therefore, since the L-shaped algorithm is applied on the original problem, one can also apply the same algorithm on the local branching subproblems. In [17], the local branching subproblems are solved until an $\epsilon - opt$ solution was found. To obtain the lower bounds $\overline{\Theta}_n$, $n = 1, \ldots, m$, one may use the same approach. Subproblems $P_n$, $n = 1, \ldots, m$, can be solved using the L-shaped algorithm until an $\epsilon - opt$ solution is found. The lower bound on recourse can then be derived from the general lower bound $\underline{z}$. In this case, through the use of local branching, one is exploring the feasible region of the original problem (1)-(5).

There is another way of using (19)-(25), which is to generate them locally for each of the subproblems explored by the 0-1 integer L-shaped algorithm. In step 2 of the algorithm, one may either apply a local branching descent to cut a solution for which the separation routine is unable to find a feasibility cut that is violated or to tighten the lower bound $c^\top x^\nu + \Theta^\nu$ associated with the subproblem considered. By working locally on the subproblems of the 0-1 integer L-shaped algorithm, one reduces considerably the size of subproblems $P_n$, $n = 1, \ldots, m$, making it easier to find better quality lower bounds. The drawback of such an approach is that the inequalities that are found are only valid locally. However, if one is able to obtain better lower bounds for each of the subproblems explored by the L-shaped algorithm, then one obtains a better general lower bound $\underline{z}$. Furthermore, by increasing values $c^\top x^\nu + \Theta^\nu$, one is able to fathom the subproblems at an earlier stage. By reducing the number of active subproblems, one can accelerate the convergence of the algorithm. These principles will now be applied to the case of the single vehicle routing problem with stochastic demands.

# 4 The single vehicle routing problem with stochastic demands

To present the single vehicle routing problem with stochastic demands, let us first define an undirected graph $G(V, E)$, 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 where a vehicle must start and finish a route. This route must visit each of the customers (i,e., $V \setminus \{v_1\}$) once while minimizing the total travel cost. Up until now, this problem represents the classical traveling salesman problem. However, the single vehicle routing problem with stochastic demands has two additional characteristics. The vehicle has a limited capacity $D$ and each of the customers $j \in V \setminus \{v_1\}$ has a nonnegative demand that may be stochastic. Numerical tests will be conducted on instances where each client $j \in V \setminus \{v_1\}$ has a demand that is a Normal random variable (i.e., $\xi_j \sim N(\mu_j, \sigma_j)$, truncated at zero) and all demands are independently distributed.

In this case, whenever the vehicle serves a certain client, there is always the risk that the observed demand exceeds the residual capacity. When such 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. One makes the hypothesis that the stochastic demands are unknown when routing decisions are being made. Therefore, the problem has two stages. One must first construct a route and then the cost of recourse may be determined. The optimization problem consists of finding a solution that minimizes simultaneously the original

travel cost and the expected cost of recourse. The model is therefore defined as follows:

$$\text{Min} \quad \sum_{i<j} c_{ij}x_{ij} + \mathcal{Q}(x) \tag{44}$$

$$\text{s.t.} \quad \sum_{j=2}^{n} x_{1j} = 2, \tag{45}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2, \ k = 2, \ldots, n, \tag{46}$$

$$\sum_{i\in S} \sum_{\substack{j\notin S \\ j>i}} x_{ij} + \sum_{i\notin S} \sum_{\substack{j\in S \\ j>i}} x_{ij} \geq 2, \ S \subseteq V, \ |S| \geq 3, \tag{47}$$

$$x_{ij} \in \{0,1\}, \ 1 \leq i < j \leq n. \tag{48}$$

An important point to be made concerns the orientation of the routes that are considered. In a deterministic context, this concept is irrelevant since the cost of the solution remains the same for both orientations. However, in the stochastic case, because of the cost of recourse, the value of a solution depends on the orientation chosen. Since demands become known only when customers are visited, one must decide, a priori, which orientation to use. Therefore, the recourse function is: $\mathcal{Q}(x) = Min\{\mathcal{Q}^1(x), \mathcal{Q}^2(x)\}$, where $\mathcal{Q}^1(x)$ and $\mathcal{Q}^2(x)$ correspond to the recourse cost for each of the two orientations. Let us define the $r-th$ feasible solution $x^r$ by vector $V^r = (v_{r_1} = v_1, v_{r_2}, \ldots, v_{r_{n+1}} = v_1)$. If the goods to be delivered (or collected) are divisible, then one may measure the recourse of solution $x^r$ for the first orientation in the following way (see [14]):

$$\mathcal{Q}^1(x^r) = 2\sum_{j=2}^{n}\sum_{l=1}^{\infty} P\left(\sum_{s=2}^{j-1}\xi_{r_s} \leq lD < \sum_{s=2}^{j}\xi_{r_s}\right) c_{1r_j}.$$

The probability term is associated with the event of having the $l$th failure occur at customer $r_j$. The computation of $\mathcal{Q}^2(x^r)$ follows the same principles, one only needs to replace the index of $\xi$ by $r_{n+2-s}$.

To solve (44)-(48) using the 0-1 integer L-shaped algorithm, one must first relax constraints (47) and (48) and then approximate the recourse function $\mathcal{Q}(x)$ using $\Theta$. Constraints of type (47) are added in step **(2.3)** of the algorithm and one may use optimality cuts of type (11) or (12), whenever a feasible solution is obtained.

Gendreau and *al.* [9] were the first to apply the standard L-shaped algorithm to the single vehicle routing problem with stochastic demands. In 1999, Hjorring and Holt [12] proposed a new type of cut that uses information taken from partial routes for problem (44)-(48). A partial route is made up of three sets. Using the notation proposed by Laporte and *al.* [14], let us first define the

two ordered sets $S = \{v_1, \ldots, v_s\}$ and $T = \{v_1, \ldots, v_t\}$. 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 $(v_1, \ldots, v_s)$ and $(v_t, \ldots, v_1)$ which define the beginning and end 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, then the following inequality is valid for problem (44)-(48):

$$\Theta \geq L + (Q - L)W(x). \tag{49}$$

To obtain a lower bound $Q$, Hjorring and Holt [12] propose to create an artificial vertex $v_0$ for which the demand is $\xi_0 = \sum_{v_i \in U\setminus\{v_s,v_t\}} \xi_i$ and the cost of return to the depot is $c_{10} = \min_{v_i\in U\setminus\{v_s,v_t\}} \{c_{1i}\}$. By using cost $c_{10}$, one can calculate a lower bound on the partial route by measuring the value of recourse for $(v_1, \ldots, v_s, v_0, v_t, \ldots, v_1)$. As for the general lower bound $L$, one can use the same technique as the one that Laporte and *al.* [14] propose for the case of the general stochastic vehicle routing problem.

Inequalities (49) may be added in step **(2.3)** of the 0-1 integer L-shaped algorithm since one can generate them using an integer or continuous solution $x^\nu$. There is an important point to be made concerning the separation algorithm needed to find violated inequalities of type (49). Let us consider $(x^\nu, \Theta^\nu)$, a solution to a *current problem* solved by the L-shaped algorithm. Vector $x^\nu$ is either integer or continuous. Since there is only one vehicle in problem (44)-(48), one can always construct a partial route using $x^\nu$ for which $W(x^\nu) = 1$. Therefore, one obtains a violated inequality (49) using this partial route if $\Theta^\nu < Q$.

Let us now consider how local branching may be used in the case of the single vehicle routing problem with stochastic demands. One may start a local branching descent from any solution $x^\nu$ provided by the L-shaped algorithm. However, one needs an integer vector to create the first neighbourhood to explore. If $x^\nu$ is integer, then one may use the vector directly. If $x^\nu$ is continuous, then by rounding to the nearest integer each component of $x^\nu$, one obtains an integer vector $\hat{x}^\nu$ which, considering the Hamming distance, is the nearest to $x^\nu$. In this case, vector $\hat{x}^\nu$ will define the starting point of the local branching descent. There is an important point to be made when one considers vector $\hat{x}^\nu$:

**Proposition 4.1.** *Let $\hat{x}$ represent a feasible route, then constraint $\Delta(x, \hat{x}) \geq 4$ is a valid inequality for problem (13)-(18) in the case of the single vehicle routing problem with stochastic demands.*

*Proof.* The closest feasible solutions to $\hat{x}$ are in the $2-opt$ neighbourhood. The $2-opt$ neighbourhood of solution $\hat{x}$ is obtained by deleting two edges from route $\hat{x}$ and replacing them by two other edges that were not in route $\hat{x}$ in order to form a new feasible solution. To do so, one must have at least two $(i,j)$ for which $\hat{x}_{ij} = 1 \rightarrow 0$ and at least two $(i,j) \in E$ for which $\hat{x}_{ij} = 0 \rightarrow 1$. Therefore, all solution in the $2-opt$ neighbourhood of $\hat{x}$ lie at a Hamming distance of 4 from $\hat{x}$. □

By rounding components of $x^\nu$, one may obtain a feasible or infeasible integer vector $\hat{x}^\nu$. If $\hat{x}^\nu$ is feasible and $\Delta(x^\nu, \hat{x}^\nu) < 4$, then by adding constraint $\Delta(x, \hat{x}) \geq 4$ one is able to cut off solution $x^\nu$ and continue the separation process. Therefore, a local branching descent need not be performed if the previous condition is verified. Only when $\hat{x}^\nu$ is either infeasible or feasible but $\Delta(x^\nu, \hat{x}^\nu) \geq 4$, is a local branching descent applied using as starting point vector $\hat{x}^\nu$. One must now determine how to perform the local branching descent. There are three points to be adressed, one must first decide how to calculate the lower bounds needed, then one must find in each of the subproblems explored a new integer vector that will be used in the branching scheme and finally, one must decide how the branching decision will be taken.

The *current problem* (13)-(18) is used in order to obtain the lower bounds for the local branching subproblems $P_n$ $(n = 1, \ldots, m)$ that are explored. One may use problem (13)-(18) at some point in the search tree of the 0-1 integer L-shaped algorithm to find local lower bounds. By doing so, the valid inequalities obtained can only be used locally on the subproblem considered. Another possibility, is to use problem (13)-(18) directly, making it possible to create general valid inequalities. Both strategies will be applied to obtain the computational results in the next section. To obtain values $\overline{\Theta}_n$ for $n = 1, \ldots, m$, the integrality constraints are temporarily reintroduced in (13)-(18). The separation algorithms are then used to improve the lower bounds in the local branching subregions that are explored. By doing so, one is using the L-shaped algorithm itself to increase the values obtained for $\overline{\Theta}_n$, $n = 1, \ldots, m$. To limit the effort spent improving the lower bounds, a maximum number of calls to the separation algorithms is imposed. The maximum number of calls was set to three to obtain the computational results in the next section. If for a particular subregion thus explored, the associated subproblem $P_n$ is solved exactly, then one obtains a feasible route $x_n$ that is optimal in the subregion defined by $P_n$ (i.e., $\mathcal{Q}(x_n) \leq \mathcal{Q}(x)\ \forall x \in P_n$). In this case, and one may set $\overline{\Theta}_n = \mathcal{Q}(x_n)$. Otherwise, the lower bound $\overline{\Theta}_n$ is simply derived from $\underline{z}_n = c^\top x_n + \Theta_n$. By proceeding in this way, one also obtains the integer vector (i.e., $x_n$) needed for the branching scheme.

Originally in [17], local branching is used to search for different optimality cuts, using as starting point the optimal solution to the Benders master problem. By doing so, one can simultaneously work on improving the upper and lower

bounds used in the Benders decomposition algorithm. By using local branching in the L-shaped algorithm, one may hope to obtain the same benefits. Since the local branching descent may provide new feasible solutions, one can find better upper bounds. By adding the local branching valid inequalities to the subproblems generated by the L-shaped algorithm, one can improve the general lower bound. However, to keep the procedure effective, one must control the effort spent generating the local branching descent. It was found that to better explore the feasible region of problems (13)-(18), one should always branch using solutions $x_n$ $(n = 1, \ldots, m)$ regardless of the quality of the lower bounds obtained. Therefore, the size of the neighbourhood is fixed to a certain value $\kappa$ which remains unchanged. Furthermore, the effort spent on each descent will also be controled by fixing value $m$ as to be the maximum depth allowed.

One can now conclude this section by describing how all these valid inequalities are generated at the separation step of the 0-1 integer L-shaped algorithm. Step **(2.3)** starts by a search of violated subtour elimination constraints. The heuristics that are used to find these cuts, are the ones that Lysgaard and *al.* [15] present for capacity inequalities. These procedures were obtained from the CVRPSEP package proposed by the authors. Subtour elimination constraints are added as long as the heuristics are able to find them. When these heuristics fail, separation continues by searching for violated inequalities of type (49). If a violated cut is found, one restarts the separation step by searching for subtour elimination constraints. As is shown by Hjorring and Holt [12], inequalities of type (49) are necessary to solve efficiently the single vehicle routing problem with stochastic demands. Since these inequalities provide useful information concerning the value of recourse, by using them, one prevents against the risk of enumeration. When one is unable to cut the current solution using either constraints (47) or inequalities of type (49), then a local branching descent is performed using fixed values $\kappa$ and $m$. Local branching is used once to tighten the lower bound associated with the active subproblem. At the end of the separation step, the local branching system (19)-(25) is added to (13)-(18) and the subproblem is solved once using the integrality constraints. In doing so, we obtain the lower bound $c^\top x^\nu + \Theta^\nu$. The solution process can then proceed as before.

# 5 Computational results

To properly test the different ideas proposed in this paper a series of test problems were generated. The problem generator used follows the same principles as the one proposed in [12]. Therefore, the graph vertices were generated in a $[0, 100]^2$ square following uniform distributions and the cost matrix was then set to be the Euclidean distances between vertices. Each customer was assigned an average demand following a $[1, 10]$ uniform distribution and the stan-

dard deviation was set to be 30% of the mean. As in [12], problems of sizes $n = 20,\ 30, \ldots,\ 90$ were created. For each size, five instances were generated for which $\overline{f} = 95\%,\ 97.5\%, \ldots,\ 110\%$ (where $\overline{f} = \sum_{i=1}^{n} \mu_i / D$), for a total of 280 instances.

The first algorithm implemented is the standard L-shaped algorithm for which the partial route cuts of Hjorring and Holt [12] are added. This is the benchmark on which the implementations using local branching are compared. Using local branching, both cut generation strategies are implemented. The LB implementation refers to the case were cuts are generated locally. As for the LB1 implementation, it refers to the case were cuts are generated globally. Parameter $m$ is fixed to three on all runs. As for the size of the local branching neighbourhoods, runs are made with $\kappa = 4,\ 6,\ 8$. All three algorithms are implemented using the branching package proposed by Gendron and *al.* [10]. All experiments are performed on a 2.4 GHz AMD Opteron 64 bit processor. A maximum time of 1200 seconds is imposed on all runs (as was also imposed by Hjorring and Holt [12]) and the optimality gap considered is $\epsilon = 1\%$.

To measure the basic tradeoff in using local branching cuts, the standard implementation is first compared to the LB implementation, for which $\kappa = 4$ (LB-4). The average solution times and average gap are reported for these two implementations in Table 1. Results are aggregated in the following way: $(< 60)$ refers to those instances that are solved in less than 60 seconds by the standard algorithm, $([60, 1200])$ to those that take between 60 and 1200 seconds and $(> 1200)$ includes those instances that the standard implementation is unable to solve in the maximum alotted time. As for the undefined (*und.*) line, it refers to instances for which the standard algorithm is unable to find a feasible solution after 1200 seconds of computation time. Whenever LB-4 is either able to solve, or obtain a feasible solution to, an instance for which the standard algorithm fails, then the (*sol.*) line is used. The (*not sol.*) line refers to those instances for which both implementations failed. All times reported are in seconds. Whenever the implementations are unable to solve the instances in the maximum time allowed, then the average gap obtained is given.

When comparing the standard and LB-4 implementations, one first observes that the results for LB-4 are significantly better. The L-shaped algorithm is either faster or obtains better results when one uses the local branching cuts. The only exception is the case of the easy instances $(< 60)$ for the problems of size $n = 60$, for which the average time of LB-4 is 6.71 seconds compared to 5.69 seconds for the standard implementation, and the problems of size $n = 90$, for which the times are 8.97 seconds compared to 6.14 seconds. In the case of the medium instances $([60, 1200])$, which corresponds to a total of 30 problems, the sum of average times for the standard implementation is 2603.8 seconds compared to 183.38 seconds for LB-4. This corresponds to a reduction in solution times by a factor that is slightly over 14. Furthermore, 24 of the

| n | Times | nb. | Standard | LB-4 |
|---|---|---|---|---|
| 20 | $< 60$ | 35 | 0.15 | 0.1 |
| 30 | $< 60$ | 31 | 2.58 | 0.83 |
| | $[60, 1200]$ | 1 | 134.02 | 1.53 |
| | $> 1200$ *sol.* | 3 | 3.05% | 359.27 |
| 40 | $< 60$ | 30 | 4.93 | 1.79 |
| | $[60, 1200]$ | 3 | 578.67 | 28.44 |
| | $> 1200$ *sol.* | 1 | 2.95% | 136.32 |
| | $> 1200$ *not sol.* | 1 | 3.00% | 1.37% |
| 50 | $< 60$ | 26 | 5.95 | 2.25 |
| | $[60, 1200]$ | 2 | 578.67 | 8.92 |
| | $> 1200$ *sol.* | 2 | 2.15% | 109.36 |
| | $> 1200$ *not sol.* | 5 | 2.82% | 1.59% |
| 60 | $< 60$ | 16 | 5.69 | 6.71 |
| | $[60, 1200]$ | 5 | 450.16 | 30.16 |
| | $> 1200$ *sol.* | 6 | 2.41% | 528.05 |
| | $> 1200$ *not sol.* | 8 | 3.58% | 1.93% |
| 70 | $< 60$ | 14 | 3.51 | 2.55 |
| | $[60, 1200]$ | 6 | 200.4 | 13.48 |
| | $> 1200$ *sol.* | 3 | 1.90% | 157.34 |
| | $> 1200$ *not sol.* | 10 | 3.78% | 2.10% |
| | *und. sol.* | 2 | - | 369.69 |
| 80 | $< 60$ | 11 | 10.91 | 8.21 |
| | $[60, 1200]$ | 9 | 392.99 | 64.15 |
| | $> 1200$ *sol.* | 5 | 1.63% | 67.31 |
| | $> 1200$ *not sol.* | 6 | 2.33% | 1.46% |
| | *und. sol.* | 3 | - | 4.84% |
| | *und. not sol.* | 1 | - | - |
| 90 | $< 60$ | 13 | 6.14 | 8.97 |
| | $[60, 1200]$ | 4 | 268.89 | 36.7 |
| | $> 1200$ *sol.* | 2 | 1.21% | 595.47 |
| | $> 1200$ *not sol.* | 12 | 3.85% | 2.59% |
| | *und. sol.* | 1 | - | 2.02% |
| | *und. not sol.* | 3 | - | - |

Table 1: Solution times and average gap: Standard vs. LB-4

instances that the standard implementation is unable to solve in the maximum time allowed, are solved by LB-4. These instances are sometimes solved quite efficiently by LB-4, as is the case for problems of size $n = 80$, where five of the ($> 1200$) instances, for which the average gap obtained by the standard algorithm is 1.63%, are solved, on average, in a little more than one minute (67.31 seconds) by LB-4. Also, out of the 24 hard instances solved by LB-4, two are in the undefined (*und.*) category (i.e., the problems of size $n = 70$). Finally, when both implementations are unable to solve the problems, the average gap obtained with LB-4 is systematically lower than with the standard algorithm.

These results may be explained when one examines the separation algorithm used in the LB implementation. In all cases, the separation process starts by searching for violated subtour elimination constraints. At this point, one is trying to establish first stage feasible solutions. If the separation heuristics fail, then the algorithm turns to the recourse value to try to cut the current solution. Cuts

of type (49) are used as much as possible to better express the recourse function. When one is no longer able to generate a cut based on a partial solution, then in the standard implementation, the algorithm must branch. However, in the LB implementation, a local branching descent is applied. The lower bounds derived from the neighbourhoods explored during the local branching descent tend to be tighter. Therefore, by using the local branching inequalities, one can better express the recourse value around a solution that the partial route inequalities were unable to cut. In this case, system (19)-(25) is used to complement the cuts of type (49) that were found for the subproblem being separated. The overall benefit of this approach is that the lower bounds obtained for each subproblem created by the algorithm are tighter. Furthermore, since the local branching search may provide new feasible solutions, one is also able to find better upper bounds at an earlier stage. In turn, both of these benefits tend to make the solution process much faster.

| n | Times | Standard | LB-4 | LB-6 | LB-8 |
|---|---|---|---|---|---|
| 20 | $< 60$ | 35 | 35 | 35 | 35 |
| 30 | $< 60$ | 31 | 32 | 33 | 32 |
| | $[60, 1200]$ | 1 | 3 | 1 | 2 |
| | $> 1200$ | 3 | 0 | 1 | 1 |
| 40 | $< 60$ | 30 | 33 | 33 | 34 |
| | $[60, 1200]$ | 3 | 1 | 1 | 0 |
| | $> 1200$ | 2 | 1 | 1 | 1 |
| 50 | $< 60$ | 26 | 29 | 29 | 29 |
| | $[60, 1200]$ | 2 | 1 | 2 | 2 |
| | $> 1200$ | 7 | 5 | 4 | 4 |
| 60 | $< 60$ | 16 | 20 | 20 | 20 |
| | $[60, 1200]$ | 5 | 7 | 8 | 7 |
| | $> 1200$ | 14 | 8 | 7 | 8 |
| 70 | $< 60$ | 14 | 22 | 21 | 21 |
| | $[60, 1200]$ | 6 | 3 | 4 | 4 |
| | $> 1200$ | 13 | 10 | 10 | 10 |
| | *und.* | 2 | 0 | 0 | 0 |
| 80 | $< 60$ | 11 | 18 | 17 | 18 |
| | $[60, 1200]$ | 9 | 7 | 8 | 7 |
| | $> 1200$ | 11 | 9 | 10 | 10 |
| | *und.* | 4 | 1 | 0 | 0 |
| 90 | $< 60$ | 13 | 16 | 16 | 16 |
| | $[60, 1200]$ | 4 | 3 | 2 | 4 |
| | $> 1200$ | 14 | 13 | 14 | 13 |
| | *und.* | 4 | 3 | 3 | 2 |

Table 2: Instances solved: standard vs. LB

Let us now examine the results obtained by both cut strategies when one increases parameter $\kappa$. To do so, Tables 2 and 3 give us the number of instances aggregated according to the previous classification but for all implementations (standard, LB and LB1) and for all runs ($\kappa = 4,\ 6,\ 8$). If one sums the results reported for all lines ($< 60$) and ($[60, 1200]$), then one obtains the total number on instances that were solved before the time limit was reached for all implementations on all runs. When examining Table 2, one may see that

the standard implementation was able to solve 204 instances compared to 230 for LB-4 and LB-6 and 231 for LB-8. Results are almost identical when one increases the value of $\kappa$ in the case of LB. By generating the cuts locally, it seems that one does not need to search large neighbourhoods to obtain good results. With LB-4, one seems to obtain the best tradeoff between the quality of results and computational effort.

| n | Times | Standard | LB1-4 | LB1-6 | LB1-8 |
|---|---|---|---|---|---|
| 20 | $< 60$ | 35 | 35 | 35 | 35 |
| 30 | $< 60$ | 31 | 32 | 31 | 31 |
| | $[60, 1200]$ | 1 | 2 | 3 | 2 |
| | $> 1200$ | 3 | 1 | 1 | 2 |
| 40 | $< 60$ | 30 | 31 | 31 | 30 |
| | $[60, 1200]$ | 3 | 2 | 3 | 4 |
| | $> 1200$ | 2 | 2 | 1 | 1 |
| 50 | $< 60$ | 26 | 25 | 27 | 26 |
| | $[60, 1200]$ | 2 | 5 | 4 | 4 |
| | $> 1200$ | 7 | 5 | 4 | 5 |
| 60 | $< 60$ | 16 | 19 | 19 | 19 |
| | $[60, 1200]$ | 5 | 3 | 3 | 3 |
| | $> 1200$ | 14 | 13 | 13 | 13 |
| 70 | $< 60$ | 14 | 19 | 17 | 19 |
| | $[60, 1200]$ | 6 | 4 | 7 | 5 |
| | $> 1200$ | 13 | 12 | 11 | 11 |
| | *und.* | 2 | 0 | 0 | 0 |
| 80 | $< 60$ | 11 | 16 | 13 | 11 |
| | $[60, 1200]$ | 9 | 9 | 12 | 13 |
| | $> 1200$ | 11 | 9 | 9 | 10 |
| | *und.* | 4 | 1 | 1 | 1 |
| 90 | $< 60$ | 13 | 15 | 15 | 12 |
| | $[60, 1200]$ | 4 | 3 | 3 | 7 |
| | $> 1200$ | 14 | 14 | 14 | 14 |
| | *und.* | 4 | 3 | 3 | 2 |

Table 3: Instances solved: standard vs. LB1

In the case of LB1, the situation is different. When analyzing Table 3, one may see that 217 instances are solved by LB1-4, 223 by LB1-6 and 221 by LB1-8. When one increases the size of $\kappa$, one is able to solve more problems in the maximum time allowed. In this case, the LB1-6 run seems to give the best results. By using larger neighbourhoods, the cuts will bound the recourse value for larger portions of the feasible region of the *current problem*. Therefore, since the LB1 implementation generates the cuts globally (cuts are reused), larger neighbourhoods tend to make the cuts useful on a larger number of subproblems explored in the search tree.

When comparing LB to LB1, one observes that by generating the cuts locally, one is able to solve a greater number of instances (230 for LB-4 compared to 223 for LB1-6). The explanation for this resides in the observation that by reusing the cuts, as is done in LB1, one makes the last subproblem processed in the separation procedure a lot harder to solve. Therefore, for a fixed max-

imum computation time, the LB1 implementation explores a smaller region of the search tree compared to the LB implementation. This proves to be a drawback on certain instances. However, when both strategies are compared to the standard algorithm, it is always preferable to apply the local branching cuts.

This section will now be concluded by analyzing how the results of both the LB and LB1 implementations vary for hard instances when the maximum time allowed is increased. To do so, a subset of hard instances was chosen (the hard instances for $n = 70,\ 80,\ 90$). Results obtained by LB-4 and LB1-6 will be compared when the maximum time allowed is increased to 2400, 3600, 4800 and 6000 seconds. In Table 4, results obtained on the 48 hard instances used are classified in three columns: the instances solved in the maximum time allowed (*sol.*), the instances that can not be solved but for which a feasible solution is obtained (*not sol.*) and the instances for which the algorithms are unable to find at least one feasible solution (*und.*). Both the number of instances and average results in seconds (for (*sol.*)), or percentage of gap (for (*not sol.*)) are given in Table 4.

| | LB | | | LB1 | | |
|---|---|---|---|---|---|---|
| Max. Time | *sol.* | *not sol.* | *und.* | *sol.* | *not sol.* | *und.* |
| 1200 | 12/228.24 | 32/2.42% | 4 | 10/275.04 | 34/3.00% | 4 |
| 2400 | 15/595.33 | 31/2.49% | 2 | 10/275.04 | 35/2.98% | 3 |
| 3600 | 17/884.89 | 29/2.44% | 2 | 12/751.01 | 34/2.93% | 2 |
| 4800 | 18/1078.25 | 29/2.52% | 1 | 12/751.01 | 34/2.87% | 2 |
| 6000 | 18/1078.25 | 29/2.45% | 1 | 12/751.01 | 34/2.84% | 2 |

Table 4: Results on hard instances: LB vs. LB1

Once again, the LB algorithm seems to outperform LB1. As the maximum computation time is increased, one may see that LB-4 is able to solve more instances compared to LB1. For 6000 seconds of computation time, LB-4 solves 18 instances compared to 12 for LB1-6. Furthermore, if one considers the (*not sol.*) column, one may see that the average gap obtained is slightly better for LB-4. Again, these results seem to show that the local branching cuts are better used locally to the subproblems explored by the L-shaped algorithm. The LB algorithm explores a larger portion of the search tree which seems to produce better results on the problems that are considered here.

# 6 Conclusion

In this paper, a new type of valid inequalities is proposed for the 0-1 integer L-shaped algorithm. These inequalities are based on the information gathered through the use of local branching descents. They provide a broader description on the value of recourse compared to the classical optimality cut. These inequalities can be implemented in the L-shaped algorithm using two different

cut generation strategies (local and global). Numerical tests are conducted on a series of single vehicle routing problems with stochastic demands. Results show that there is a clear advantage in using these new inequalities. An interesting avenue of research would be to generalize the approach proposed here to the case of general vehicle routing problems with stochastic demands. It would also be interesting to find new separation strategies using these local branching cuts.

# References

[1] S. Ahmed, M. Tawarmalani, and N.V. Sahinidis. A finite branch and bound algorithm for two-stage stochastic integer programs. *Mathematical Programming*, 100:355–377, 2004.

[2] I.L. Averbakh. An iterative method of solving two-stage discrete stochastic programming problems with additively separable variables. *USSR Computational Mathematics and Mathematical Physics*, 31(6):21–27, 1991.

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

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

[5] C.C. Caroe and R. Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24:37–45, 1999.

[6] C.C. Caroe and J. Tind. A cutting-plane approach to mixed 0-1 stochastic integer programs. *European Journal of Operational Research*, 101:306–316, 1997.

[7] C.C. Caroe and J. Tind. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83:451–464, 1998.

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

[9] 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.

[10] B. Gendron, T.G. Crainic, A. Frangioni, and F. Guertin. Oobb: Object-oriented tools for parallel branch-and-bound. In *PAREO, Mont-Tremblant, Canada*, 16–21 january 2005.

[11] R. Hemmecke and R. Schultz. Decomposition of test sets in stochastic integer programming. *Mathematical Programming*, 94:323–341, 2003.

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

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

[14] 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.

[15] 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.

[16] V.I. Norkin, Y.M. Ermoliev, and A. Ruszczyǹski. On optimal allocation of indivisibles under uncertainty. *Operations Research*, 46:381–395, 1998.

[17] W. Rei, J.F Cordeau, M. Gendreau, and P. Soriano. Accelerating benders decomposition by local branching. Technical report, Center for Research on Transportation, C.P. 6128, succursale Centre-ville Montréal QC H3C 3J7 Canada, January 2006.

[18] R. Schultz, L. Stougie, and M.H. van der Vlerk. Solving stochastic programs with integer recourse by enumeration: A framework using grobner basis reductions. *Mathematical Programming*, 83:229–252, 1998.

[19] S. Sen and J.L. Higle. The $c^3$ theorem and a $d^2$ algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming*, 104:1–20, 2005.

[20] S. Sen and H.D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106:203–223, 2006.

[21] H.D. Sherali and B.M.P. Fraticelli. A modification of benders decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization*, 22:319–342, 2002.

[22] 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.

[23] R.D. Wollmer. Two stage linear programming under uncertainty with 0-1 integer first stage variables. *Mathematical Programming*, 19:279–288, 1980.