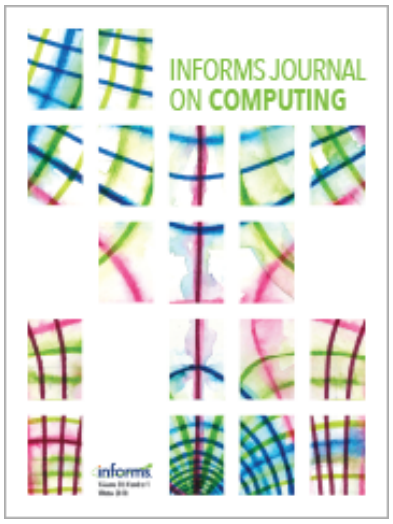

INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Accelerating Benders Decomposition by Local Branching

Walter Rei, Jean-François Cordeau, Michel Gendreau, Patrick Soriano,

To cite this article:
Walter Rei, Jean-François Cordeau, Michel Gendreau, Patrick Soriano, (2009) Accelerating Benders Decomposition by Local Branching. INFORMS Journal on Computing 21(2):333-345. https://doi.org/10.1287/ijoc.1080.0296

INFORMS Journal on Computing
Vol. 21, No. 2, Spring 2009, pp. 333–345
ISSN 1091-9856 | EISSN 1526-5528 | 09 | 2102 | 0333

DOI 10.1287/ijoc.1080.0296

Accelerating Benders Decomposition by Local Branching

Walter Rei
Département de management et de technologie, ESG, UQÀM, Montréal, Québec H3C 3P8, Canada, and CIRRELT, Montréal, Québec H3C 3J7, Canada
{rei.walter@uqam.ca}

Jean-François Cordeau
Service de l'enseignement de la gestion des opérations et de la logistique, HEC Montréal, Montréal, Québec H3T 2A7, Canada, and CIRRELT, Montréal, Québec H3C 3J7, Canada
{jean-francois.cordeau@hec.ca}

Michel Gendreau
Département d'informatique et de recherche opérationnelle, Université de Montréal, Montréal, Québec H3C 3J7, Canada, and CIRRELT, Montréal, Québec H3C 3J7, Canada
{michel.gendreau@cirrelt.ca}

Patrick Soriano
Service de l'enseignement des méthodes quantitatives de gestion, HEC Montréal, Montréal, Québec H3T 2A7, Canada, and CIRRELT, Montréal, Québec H3C 3J7, Canada
{patrick.soriano@hec.ca}

This paper shows how local branching can be used to accelerate the classical Benders decomposition algorithm. By applying local branching throughout the solution process, one can simultaneously improve both the lower and upper bounds. We also show how Benders feasibility cuts can be strengthened or replaced with local branching constraints. To assess the performance of the different algorithmic ideas presented in this hybrid solution approach, extensive computational experiments were performed on two families of network design problems. Numerical results clearly illustrate their benefits.

Key words: Benders decomposition; local branching; deterministic and stochastic network design; hybrid solution approach
History: Accepted by John W. Chinneck, Editor-in-Chief, on the advice of William Cook, former Area Editor for Design and Analysis of Algorithms; received November 2006; revised October 2007, June 2008; accepted June 2008. Published online in *Articles in Advance* November 13, 2008.

1. Introduction

Benders decomposition (Benders 1962) was introduced in the early 1960s as a solution strategy for mixed-integer problems. As shown by Geoffrion (1970a, b), this method can be classified as a pattern of projection, outer linearization, and relaxation. Let us consider the following general class of mixed 0-1 linear programming problems:

$$\min \quad c_1^\top x + c_2^\top y \tag{1}$$

$$\text{s.t.} \quad A_1 x = b_1, \tag{2}$$

$$A_2 x + E y = b_2, \tag{3}$$

$$x \in \{0,1\}^{n_1}, \tag{4}$$

$$y \in \mathbb{R}_+^{n_2}, \tag{5}$$

where $c_1 \in \mathbb{R}^{n_1}$, $c_2 \in \mathbb{R}^{n_2}$, $b_1 \in \mathbb{R}^{m_1}$, $b_2 \in \mathbb{R}^{m_2}$, $A_i \in \mathbb{R}^{n_1 \times m_i}$ $(i = 1, 2)$, and $E \in \mathbb{R}^{n_2 \times m_2}$. Although the ideas proposed in this paper can be generalized to the case where a subset of the x variables are general integers, to keep the exposition simple, only problem (1)–(5) will be considered in the following.

Projecting problem (1)–(5) onto the space defined by the binary variables x yields the problem

$$\begin{aligned} \min \quad & c_1^\top x + \inf_y \{c_2^\top y \mid Ey = b_2 - A_2 x,\ y \in \mathbb{R}_+^{n_2}\} \\ \text{s.t.} \quad & A_1 x = b_1, \\ & x \in \{0,1\}^{n_1}. \end{aligned}$$

Define $\Pi = \{\pi \in \mathbb{R}^{m_2} \mid E^\top \pi \le c_2\}$, and let Λ be the set of extreme points associated with Π and Φ be the set of extreme rays of cone $C = \{\pi \in \mathbb{R}^{m_2} \mid E^\top \pi \le 0\}$. Applying an outer linearization procedure to the function $\inf_y \{c_2^\top y \mid Ey = b_2 - A_2 x, y \in \mathbb{R}_+^{n_2}\}$, we can restate the original problem as the following equivalent master problem:

$$\min \quad c_1^\top x + \Theta \tag{6}$$

$$\text{s.t.} \quad A_1 x = b_1, \tag{7}$$

$$(b_2 - A_2x)^\top \lambda_i \le \Theta \quad \forall \lambda_i \in \Lambda, \tag{8}$$

$$(b_2 - A_2x)^\top \phi_j \le 0 \quad \forall \phi_j \in \Phi, \tag{9}$$

$$x \in \{0,1\}^{n_1}. \tag{10}$$

Constraints (8) are called *optimality cuts* because they define the objective value associated with feasible values of x. Constraints (9) are called *feasibility cuts* because they eliminate values of x for which the function $\inf_y\{c_2^\top y \mid Ey = b_2 - A_2x,\ y \in \mathbb{R}^{n_2}_+\} = +\infty$.

Because the number of constraints in sets (8) and (9) may be very large, Benders proposed a relaxation algorithm to solve problem (6)–(10). This algorithm can be stated as follows:

Relaxation Algorithm

Step 0. (Initialization)
$\nu = 0$, $t = 0$, $s = 0$.
$\underline{z} = -\infty$.
$\bar{z} = +\infty$.
Step 1. (Solve the relaxed master problem)
$\nu = \nu + 1$.
Solve

$$\min\ c_1^\top x + \Theta \tag{11}$$

$$\text{s.t. } A_1x = b_1, \tag{12}$$

$$(b_2 - A_2x)^\top \lambda_i \le \Theta, \quad i = 1, \dots, t, \tag{13}$$

$$(b_2 - A_2x)^\top \phi_j \le 0, \quad j = 1, \dots, s, \tag{14}$$

$$x \in \{0,1\}^{n_1}. \tag{15}$$

Let (x^ν, Θ^ν) be an optimal solution to problem (11)–(15). Set $\underline{z} = c_1^\top x^\nu + \Theta^\nu$.
Step 2. (Solve the subproblem)
Solve

$$v^\star = \max\ (b_2 - A_2x^\nu)^\top \pi \tag{16}$$

$$\text{s.t. } E^\top \pi \le c_2, \tag{17}$$

$$\pi \in \mathbb{R}^{m_2}. \tag{18}$$

If (16)–(18) is feasible:
—Set $\bar{z} = \min\{\bar{z}, c_1^\top x^\nu + v^\star\}$.
—**If** $(\bar{z} - \underline{z})/\bar{z} < \epsilon$, **Stop**.
—One obtains an extreme point λ_{t+1} violating one of the inequalities (8), which can be added to (11)–(15).
—Set $t = t + 1$ and go to Step 1.
Else:
—One obtains an extreme ray ϕ_{s+1} violating one of the inequalities (9), which can be added to (11)–(15).
—Set $s = s + 1$ and go to Step 1.

There are certain observations to be made when analyzing the Benders decomposition approach. One must first consider the fact that each time Step 1 of the relaxation algorithm is performed, an integer problem must be solved. Furthermore, this process becomes more difficult each time a new cut is added. Another important observation concerns the bounds generated by the algorithm. Because each iteration of the algorithm adds a new cut to the relaxed master problem, the lower bound $\underline{z}$ is therefore nondecreasing. However, there is no guarantee that the upper bound $\bar{z}$ is decreasing. On many instances, the evolution of the objective function value associated with the feasible solutions obtained is quite erratic.

To circumvent these difficulties, we propose to use phases of local branching (Fischetti and Lodi 2003) throughout the solution process. The aim of the local branching phase is to find better upper bounds as well as multiple optimality cuts at each iteration of the relaxation algorithm. By working simultaneously on improving the lower bound $\underline{z}$ and the upper bound $\bar{z}$, one can hope to reduce the number of relaxed master problems that need to be solved.

The remainder of this paper is organized as follows. Section 2 contains a brief review of the different techniques that have been proposed to improve the classical Benders decomposition approach. In §3, we describe the local branching strategy and how this strategy can be used to improve Benders decomposition. This is followed by computational experiments in §4. Finally, we conclude in §5 with a discussion of the performance of our approach as well as ideas for future research.

2. Related Work

Over the years, several techniques have been proposed to speed up the classical Benders decomposition approach. Research has mainly focused on either reducing the number of integer-relaxed master problems being solved or on accelerating the solution of the relaxed master problem. McDaniel and Devine (1977) proposed to generate cuts from solutions obtained by solving the linear relaxation of (11)–(15). Because any extreme point (or extreme ray) of the dual subproblem generates a valid optimality (or feasibility) cut for the integer master problem, one can hope to generate useful information for the integer case by adding the cuts derived from the continuous relaxation, therefore reducing the number of integer master problems that need to be solved. Côté and Laughton (1984) also observed that one does not have to solve the relaxed master problem to optimality in Step 1 of the relaxation algorithm. One only needs to find an integer solution to generate an optimality (or feasibility) cut. Therefore, any heuristic that is adapted for the specific problem to be solved can be used to generate different cuts. The main drawback of such a strategy is that by generating only the cuts associated with the solutions obtained using the

heuristic, one can fail to generate cuts that are necessary to ensure convergence.

Magnanti and Wong (1981) proposed to accelerate the convergence of the Benders algorithm by adding Pareto-optimal cuts. A Pareto-optimal cut is defined as follows: Consider problem (1)–(5) and let $X = \{x \mid A_1 x = b_1, x \in \{0,1\}^{n_1}\}$. A cut $(b_2 - A_2 x)^\top \lambda_1 \le \Theta$ is said to dominate another cut $(b_2 - A_2 x)^\top \lambda_2 \le \Theta$ if $(b_2 - A_2 x)^\top \lambda_1 \ge (b_2 - A_2 x)^\top \lambda_2$ for all $x \in X$ with a strict inequality for at least one point in X; a cut is said to be Pareto-optimal if no other cut dominates it. Let us now consider the case where there are multiple optimal solutions to the subproblem in Step 2. In such a case, Magnanti and Wong (1981) demonstrated that one can obtain a Pareto-optimal cut by evaluating for each of the dual solutions the associated cut at a core point of set X and then choosing the one that gives the maximum value. The addition of nondominated optimality cuts to the relaxed master problems can greatly improve the quality of the lower bound obtained in the Benders relaxation algorithm. In turn, this will accelerate the convergence by reducing the total number of iterations needed to obtain an optimal solution.

Van Roy (1983) proposed a new type of decomposition (cross-decomposition) to solve hard integer problems. The main idea in cross-decomposition is to use simultaneously primal decomposition (Benders decomposition) and dual decomposition (Lagrangean relaxation). Van Roy showed that a solution to the Lagrangean subproblem can act as a possible solution to the Benders master problem and vice versa. Therefore, one can alternately solve the two subproblems to generate useful information for the master problems. To maintain convergence, one must still solve at times the Benders master problem, but much less often. Cross-decomposition can thus be interpreted as a way to speed up the classical Benders decomposition algorithm.

Recently, Codato and Fischetti (2006) proposed to use combinatorial inequalities in the context of Benders decomposition. When considering problem (1)–(5), for the cases where $c_1 \ne [0]^\top$ and $c_2 = [0]^\top$ or $c_1 = [0]^\top$ and $c_2 \ne [0]^\top$, the purpose of subproblem (16)–(18) can be reduced to testing the feasibility of the integer solutions and only feasibility cuts are added to the master problem. For these cases, using the ideas put forward by Hooker (2000), the authors make the observation that whenever the solution to the master problem x^ν is infeasible, one may find at least one set $C \subseteq \{1, \ldots, n_1\}$ such that x_j^ν, $j \in C$, is a minimal (or irreducible) infeasible subsystem of (16)–(18). Therefore, one needs to change at least one binary value of x_j^ν, $j \in C$, to eliminate the infeasibility associated with C. To impose this condition, Codato and Fichetti (2006) define the following combinatorial Benders cut: $\sum_{i \in C: x^\nu_{j(i)}=1} (1 - x_j) + \sum_{i \in C: x^\nu_{j(i)}=0} x_j \ge 1$. The authors present a separation algorithm to find minimal infeasible subsystems. Combinatorial Benders cuts are then added to the master problem in a general branch-and-cut framework. Computational experiments were performed on two classes of mixed-integer problems. Results seem to demonstrate that by adding combinatorial Benders cuts, one can considerably improve the quality of the bounds obtained for the linear programming (LP) relaxation of the master problem.

As for techniques to reduce the solution time of the relaxed master problem, research has been mostly oriented toward the use of Lagrangean relaxation. Côté and Laughton (1984) made an observation concerning the case where set $X = \{x \mid A_1 x = b_1, x \in \{0,1\}^{n_1}\}$ has a special structure that can be exploited by solution algorithms. Then, constraints (13) and (14) would prevent the use of specifically adapted methods for solving model (11)–(15). By using Lagrangean relaxation on these constraints, one can regain the special structure of the problem. For specific values of the Lagrange multipliers, the problem can be solved efficiently with a method that exploits the structure of X. However, the integer solution obtained may not be feasible for model (11)–(15). The authors propose to use subgradient optimization to modify the Lagrange multipliers and resolve the problem. Unfortunately, at the end of this process one may still not find an optimal solution to the master problem due to the duality gap that may exist. For such a case, a branching strategy must be used in the master problem to bridge the gap.

Lagrangean relaxation may seem as a good way to circumvent the difficulty of solving the master problems. However, Holmberg (1994) made an in-depth analysis on the use of such a technique. Holmberg (1994) showed that the lower bound defined by the use of Lagrangean relaxation on the master problem cannot be better than the bound obtained from the Lagrangean dual of the original problem (the dual subproblem used in cross-decomposition). This is always the case, even when all cuts are added to the Benders master problem. Therefore, one can never hope to obtain better bounds by applying Lagrangean relaxation on the master problem (even when a large number of cuts have been added) compared with using Lagrangean relaxation on the original problem. Also, when using Lagrangean relaxation on the master problem, a lack of controllability on the integer solution obtained may prevent the approach from converging (i.e., generating necessary cuts).

3. Local Branching in Benders Decomposition

We begin this section by giving a brief description of the local branching strategy (Fischetti and Lodi 2003).

The main idea behind local branching is to divide the feasible region of a problem into smaller subregions and then use a generic solver (e.g., CPLEX) to find the best solution (or at least a good feasible solution) in each of the subregions. By doing so, one can take advantage of the fact that generic optimizers can efficiently solve small instances of a problem and then use this characteristic in a general branching strategy for solving large-scale problems.

Let us consider problem (1)–(5) and (x^0, y^0) a feasible solution to (1)–(5). By using the Hamming distance function $\Delta(x, x^0) = \sum_{j \in S_0}(1 - x_j) + \sum_{j \in N_1 \setminus S_0} x_j$ (where $N_1 = \{1, \ldots, n_1\}$ and $S_0 = \{j \in N_1 \mid x_j^0 = 1\}$), one can divide the feasible region of problem (1)–(5) by considering, on the one hand, the values of x for which $\Delta(x, x^0) \leq \kappa$ and, on the other hand, the values of x for which $\Delta(x, x^0) \geq \kappa + 1$ (where κ is a positive integer). By imposing an adequate value κ, one can solve efficiently problem (1)–(5) in the subregion defined by $\Delta(x, x^0) \leq \kappa$ using the generic optimizer. Subregion $\Delta(x, x^0) \geq \kappa + 1$ is left for further exploration. For the case where a subset of the x variables are general integers, one can either apply the distance function exclusively on the 0-1 variables or on the entire x vector. However, as shown by Fischetti and Lodi (2003), when general integer variables are used, the distance function becomes much more involved.

Let us now consider the local branching algorithm at iteration t, where (x^t, y^t) is the current feasible solution to (1)–(5). Let x^j, $j \in J^t$, be a series of values such that $\inf_y\{c_2^\top y \mid Ey = b_2 - A_2 x^j, y \in \mathbb{R}_+^{n_2}\} \neq +\infty$ (i.e., previous feasible solutions). Let us also define a set I^t such that $I^t \subseteq J^t$. By using the function $\Delta(x, x^t)$ as well as sets J^t and I^t, one can divide the unexplored feasible region of problem (1)–(5) into the following two subproblems (thus creating a left and right branch):

$$
\begin{aligned}
(P_t) \quad \min \quad & c_1^\top x + c_2^\top y \\
\text{s.t.} \quad & A_1 x = b_1, \\
& A_2 x + Ey = b_2, \\
& \Delta(x, x^j) \geq 1, \quad j \in J^t, \\
& \Delta(x, x^i) \geq \kappa_i, \quad i \in I^t, \\
& \Delta(x, x^t) \leq \kappa, \\
& x \in \{0, 1\}^{n_1}, \\
& y \in \mathbb{R}_+^{n_2}.
\end{aligned}
$$

$$
\begin{aligned}
(\bar{P}_t) \quad \min \quad & c_1^\top x + c_2^\top y \\
\text{s.t.} \quad & A_1 x = b_1, \\
& A_2 x + Ey = b_2, \\
& \Delta(x, x^j) \geq 1, \quad j \in J^t, \\
& \Delta(x, x^i) \geq \kappa_i, \quad i \in I^t, \\
& \Delta(x, x^t) \geq \kappa + 1, \\
& x \in \{0, 1\}^{n_1}, \\
& y \in \mathbb{R}_+^{n_2}.
\end{aligned}
$$

The feasible region of P_t is limited to all feasible values of (x, y) such that $x \in \{x \in \{0, 1\}^{n_1} \mid \Delta(x, x^j) \geq 1, j \in J^t, \Delta(x, x^i) \geq \kappa_i, i \in I^t\}$ and for which the Hamming distance between x and x^t is less than or equal to κ.

Subproblem P_t is solved using the generic optimizer. Let (x^{t+1}, y^{t+1}) be the optimal solution to P_t. If $c_1^\top x^{t+1} + c_2^\top y^{t+1} < c_1^\top x^t + c_2^\top y^t$, then one can set $\kappa_t = \kappa + 1$, replace constraint $\Delta(x, x^t) \leq \kappa$ with $\Delta(x, x^t) \geq \kappa_t$, set $I^{t+1} = I^t \cup \{t\}$ (which gives subproblem $\bar{P}_t$), and then divide the feasible region of $\bar{P}_t$ as before using the function $\Delta(x, x^{t+1})$ (creating subproblems P_{t+1} and $\bar{P}_{t+1}$). One can then proceed by solving the new left branch created (subproblem P_{t+1}). However, if $c_1^\top x^{t+1} + c_2^\top y^{t+1} \geq c_1^\top x^t + c_2^\top y^t$ or P_t is infeasible, a diversification procedure is applied. The diversification procedure consists of replacing $\Delta(x, x^t) \leq \kappa$ with $\Delta(x, x^t) \leq \kappa + 1$ in subproblem P_t and then adding constraint $\Delta(x, x^t) \geq 1$ (with $J^{t+1} = J^t \cup \{t\}$). Therefore, one increases the size of the feasible region of P_t and, by adding $\Delta(x, x^t) \geq 1$, eliminates solution (x^t, y^t). Subproblem P_t is solved again, and the procedure continues by testing the new solution obtained. For the diversification procedure, it should be noted that Fischetti and Lodi (2003) proposed to replace $\Delta(x, x^t) \leq \kappa$ with $\Delta(x, x^\nu) \leq \kappa + \lceil \kappa/2 \rceil$. However, for the purpose of this paper, in an effort to limit the increase in the size of the active subproblem, an increase of one is used.

An important observation to be made concerns the solution of the active subproblem. Because the size of the neighborhood defined by the Hamming distance may become large, it may be impossible to solve the subproblem to optimality. Fischetti and Lodi (2003) proposed a series of techniques to circumvent this difficulty. These techniques include imposing a time limit on the solution of the active subproblem as well as a series of diversification mechanisms derived from local search metaheuristics. A complete description of these techniques is provided in Fischetti and Lodi (2003). For the purpose of this paper, it should be specified that every active subproblem will be solved until an $\bar{\epsilon} - opt$ solution is found or a predetermined time limit is reached.

Let us now examine the usefulness of local branching in the context of Benders decomposition. By fixing a time limit and a starting feasible solution (x^0, y^0), the local branching algorithm searches for a better upper bound in a neighbourhood around (x^0, y^0). Let $P_0, P_1, \ldots, P_L$ be the series of subproblems solved by

the algorithm. Without loss of generality, let us suppose that the first L' ($L' \leq L$) of these subproblems are feasible and let $(x^1, y^1), (x^2, y^2), \ldots, (x^{L'}, y^{L'})$ be the solutions obtained for each of these subproblems. Because the algorithm adds a constraint $\Delta(x, x^\nu) \geq 1$ each time the value of the solution obtained is not better than the previous one, then at least $\lceil (L'+1)/2 \rceil$ of the solutions $(x^0, y^0), (x^1, y^1), \ldots, (x^{L'}, y^{L'})$ will be different.

In the context of Benders decomposition, let us now suppose that at iteration ν of the relaxation algorithm, one uses the solution to the relaxed master problem x^ν as a starting point for a phase of local branching. It is important to observe that x^ν does not have to be a feasible solution (i.e., induce a feasible subproblem (16)–(18)) to be used as a starting point for the local branching algorithm. One can start from an infeasible point and search its neighbourhood for feasible solutions. At the end of the local branching phase, one obtains an upper bound $\hat{z} = \min_{l=1,\ldots,L'}\{c_1^\top x^l + c_2^\top y^l\}$ on (1)–(5) and each feasible solution identified can be used to generate an optimality cut. Therefore, with at least $\lceil L'/2 \rceil$ (or $\lceil (L'+1)/2 \rceil$ if x^ν is feasible) different solutions, one can create a pool of possible cuts to be added to the relaxed master problem (11)–(15). At iteration ν, solution x^ν provides a point in the set X that must be considered in the Benders solution process. By searching in the neighbourhood of this point, one may hope to find useful information that will help the Benders algorithm in eliminating a larger subregion of X. This strategy can also be applied during the LP phase defined by McDaniel and Devine (1977). If x^ν is a continuous solution, by rounding each of its components to the nearest integer, one obtains an integer vector that is closest to x^ν when considering the Hamming distance function. This integer vector can then be used to apply the branching scheme defined previously.

Another point to be made concerns the branching strategy used in the local branching algorithm as well as the size of the subregions explored (parameter κ). In the classical local branching algorithm, the branching decision is applied whenever the algorithm finds a better feasible solution. However, one should keep in mind the two purposes of the local branching search in the context of Benders decomposition: finding better upper bounds and generating different cuts to obtain better lower bounds. Local branching offers a general framework that makes it easy to pursue both objectives. By keeping a relatively low parameter κ and by applying the branching decision often, one is sure to quickly explore different parts of the feasible region. In this case, the emphasis is placed on finding different feasible solutions (i.e., optimality cuts). Because the main difficulty related to problems of type (1)–(5) concerns the quality of the lower bounds obtained, one can expect to accelerate the search process by generating a wide pool of optimality cuts at each iteration of the relaxation algorithm.

Not every cut identified after a local branching phase may be worth adding to problem (11)–(15). One must choose cuts that permit what Holmberg (1990) defined as cut improvement. At iteration ν of the relaxation algorithm, an optimality cut that gives cut improvement is a new cut (i.e., a cut not yet present in (11)–(15)) that may be active in an optimal solution of problem (1)–(5). In cross-decomposition, Van Roy (1983) uses the solution to the dual subproblem to generate a cut in the Benders relaxed master problem. Because there is no guarantee that this cut will be useful, Van Roy (1983) uses this next result to verify cut improvement. Let the relaxed master problem at iteration ν be

$$\begin{aligned} \min\ & c_1^\top x + \Theta \\ \text{s.t.}\ & A_1 x = b_1, \\ & (b_2 - A_2 x)^\top \lambda_i \leq \Theta, \quad i = 1, \ldots, t, \\ & (b_2 - A_2 x)^\top \phi_j \leq 0, \quad j = 1, \ldots, s, \\ & x \in \{0,1\}^{n_1}. \end{aligned}$$

Let $\tilde{x}$ be a feasible solution and $\bar{z}$ the value of the best feasible solution used to generate an optimality cut in the previous problem. If $c_1^\top \tilde{x} + (b_2 - A_2\tilde{x})^\top \lambda_i < \bar{z}$, $i = 1, \ldots, t$, then $\tilde{x}$ gives cut improvement in the relaxed master problem. This result may also be used to identify which of the different feasible solutions obtained after the local branching phase allow cut improvement.

Because each added cut will make the relaxed master problem harder to solve, one may be interested in adding only a subset of the identified cuts that allow cut improvement. In doing so, measuring the depth of each cut (i.e., the "size" of the feasible region of the relaxed master problem that is eliminated by adding the cut) is quite useful. However, defining such a measure is not trivial. Similarly to Magnanti and Wong (1981), what we propose is to use a core point to simulate the direction to the optimal solution of problem (1)–(5). By identifying a point $\hat{x}^0$ in the interior of the feasible region of (1)–(5) and then evaluating the cuts at $\hat{x}^0$, one can sort the cuts according to how many solutions of the relaxed master problem are eliminated in that direction. Using this criterion, the deepest cut is

$$\lambda \in \arg\max\{(b_2 - A_2\hat{x}^0)^\top \lambda_l \mid l = 0, \ldots, L'\}, \qquad (19)$$

where λ_l, $l = 0, \ldots, L'$, are the optimal solutions to the dual of subproblem (16)–(18) when $x = x^l$. The effectiveness of this measure will depend on the core point that is chosen. Therefore, one can either fix this point

at the beginning or adjust it iteratively throughout the solution process.

There is another important point to be made when using local branching in a Benders decomposition process. This point concerns the subproblems that are found to be infeasible. In the classical Benders decomposition approach, at iteration ν, when the solution to the relaxed master problem (x^ν) is infeasible, a feasibility cut is added to (11)–(15) using ϕ_{s+1}. This cut will eliminate from further consideration x^ν as well as all values of $x \in X$ for which $(b_2 - A_2x)^\top \phi_{s+1} > 0$. When using local branching, one will sometimes find subproblems that are infeasible. Suppose that one executes a local branching phase around x^ν. Let us consider subproblem P_l, which is defined by the local branching constraints $\Delta(x, x^j) \geq 1$, $j \in J^l$, $\Delta(x, x^i) \geq \kappa_i$, $i \in I^l$, and $\Delta(x, x^l) \leq \kappa$. If P_l is found to be infeasible, then one has found in the neighbourhood of x^ν a subregion that contains no feasible solution. Because the feasible solutions x^j, $j \in J^l$, have already been considered by the algorithm, one can use the information given by P_l by adding constraints $\Delta(x, x^i) \geq \kappa_i$, $i \in I^l$ and $\Delta(x, x^l) \geq \kappa + 1$ to (11)–(15), thus eliminating from further consideration the subregion defined by $\Delta(x, x^i) \geq \kappa_i$, $i \in I^l$ and $\Delta(x, x^l) \leq \kappa$. In this case, the local branching constraints are used in a similar way to the combinatorial Benders cuts proposed by Codato and Fischetti (2006). They can either be used as a complement to the feasibility cuts or, if a local branching phase is used each time the relaxed master problem is solved and all integer variables in the original problem are binary, as a way to replace constraints (14).

Finally, it should be noted that during a local branching phase, the series of subproblems $P_0, P_1, \ldots, P_L$ that are solved retain the same structure as the original problem (1)–(5). Therefore, if Benders decomposition can be specialized to solve problem (1)–(5), then the same specialization can also be used to solve $P_0, P_1, \ldots, P_L$. Benders decomposition has the added advantage that the cuts that are obtained when solving a subproblem can be reused for the solution of future subproblems. This is an important observation to be made: by using Benders decomposition as the optimizer in local branching, the ideas proposed in this paper can be extended to all settings where Benders decomposition is applied or could be applied.

4. Computational Experiments

The problems used to test the ideas proposed in this paper are taken from the general class of network design problems. As is apparent in the recent survey of Costa (2005), network design problems have often been solved successfully using Benders decomposition. The first problem that is considered is the deterministic multicommodity capacitated fixed-charge network design problem (MCFND). What motivated the choice of the MCFND is the fact that it offers a simple formulation that represents well this general class of design problems. Because the idea of using local branching in Benders decomposition is proposed in a general setting as a means of accelerating the Benders method, no effort was made to adapt the algorithms to this first specific problem. All results reported on the MCFND are only used to measure the general trade-offs in using local branching within the Benders relaxation algorithm. In no way should this strategy be viewed as the best way of solving the MCFND.

To obtain results on a state-of-the-art application of the Benders algorithm, a second series of tests are conducted on a stochastic integrated model for logistics network design. Benders decomposition has been shown to be quite efficient in solving many stochastic programming problems (e.g., see the L-shaped method of Van Slyke and Wets 1969 and its numerous applications). In the case of logistics network design, Santoso et al. (2005) showed how Benders decomposition is a useful tool to quickly obtain high-quality solutions when there is uncertainty in the demand. Recently, Cordeau et al. (2006) have introduced a deterministic logistics network design model that integrates location decisions with product assignment, sourcing decisions, and product flows. Cordeau et al. (2006) have also described a Benders decomposition approach that is shown to be very competitive on the problems tested. Therefore, to further validate the ideas proposed in this paper, we generated stochastic versions of the problems proposed by Cordeau et al. (2006) and adapted their Benders decomposition approach to the stochastic case. In light of the observations made on the previous tests, a specialized strategy using local branching within the Benders algorithm is proposed to accelerate the solution process. The next two subsections are used to present the results obtained on both design problems.

4.1. MCFND

The MCFND is defined as follows: Let $G(N, A, K)$ be a directed graph where N, A, and K are, respectively, the sets of nodes, arcs, and commodities that have to be routed over the network. For each arc $(i, j) \in A$, let us associate a Boolean variable x_{ij} taking value one if (i, j) is used and zero otherwise. For each $(i, j) \in A$ and $k \in K$, let variable y_{ij}^k denote the flow of commodity k on arc (i, j). Let us also define parameters f_{ij}, c_{ij}^k, u_{ij}, and d_k as being, respectively, the fixed cost of using arc (i, j), the routing cost of one unit of commodity k through arc (i, j), the total capacity of arc (i, j), and the demand for commodity k,

$\forall (i, j) \in A$ and $\forall k \in K$. The MCFND problem can then be formulated as

$$\min \sum_{(i,j)\in A} f_{ij}x_{ij} + \sum_{(i,j)\in A}\sum_{k\in K} c_{ij}^k y_{ij}^k \tag{20}$$

$$\text{s.t.} \sum_{j\in N_i^+} y_{ij}^k - \sum_{j\in N_i^-} y_{ji}^k = \begin{cases} d_k, & i = O(k), \\ 0, & i \notin \{O(k), D(k)\} \ \forall i \in N, \ \forall k \in K, \\ -d_k, & i = D(k), \end{cases} \tag{21}$$

$$\sum_{k\in K} y_{ij}^k \le u_{ij}x_{ij} \quad \forall (i,j) \in A, \tag{22}$$

$$y_{ij}^k \ge 0 \quad \forall (i,j) \in A, \ \forall k \in K, \tag{23}$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A, \tag{24}$$

where $N_i^+ = \{j \mid (i,j) \in A\}$, $N_i^- = \{j \mid (j,i) \in A\}$, and the pair $\{O(k), D(k)\}$ corresponds to the origin and destination for the transportation of commodity $k \in K$. The MCFND consists of minimizing the fixed and routing costs (20) subject to the flow conservation constraints (21) and capacity constraints (22).

Applying the Benders decomposition scheme on problem (20)–(24), one obtains the following master problem:

$$\min \sum_{(i,j)\in A} f_{ij}x_{ij} + \Theta$$

$$\text{s.t.} \sum_{(i,j)\in A} u_{ij}\pi_{ij}x_{ij} + \sum_{k\in K} d_k\alpha_{O(k)}^k - d_k\alpha_{D(k)}^k \le \Theta \quad \forall (\pi,\alpha) \in \Lambda,$$

$$\sum_{(i,j)\in A} u_{ij}\pi_{ij}x_{ij} + \sum_{k\in K} d_k\alpha_{O(k)}^k - d_k\alpha_{D(k)}^k \le 0 \quad \forall (\pi,\alpha) \in \Phi,$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in A,$$

where sets Λ and Φ are defined as before.

All experiments were performed with a standard implementation of the Benders decomposition algorithm, including the features proposed by McDaniel and Devine (1977) as well as Magnanti and Wong (1981). This implementation will be referred to as BD. It should be noted that during the LP phase, the linear relaxation of problem (20)–(24) is solved completely. Furthermore, all optimality cuts added during the LP and integer phases are Pareto-optimal cuts. As was already noted, no effort was made to specialize the BD implementation to the MCFND problem. The only exception is the addition of the following well-known cutset inequalities: $\sum_{j\in N_i^+} u_{ij}x_{ij} \ge \sum_{k\in K \mid i=O(k)} d^k \ \forall i \in N$ and $\sum_{j\in N_i^-} u_{ij}x_{ij} \ge \sum_{k\in K \mid i=D(k)} d^k \ \forall i \in N$. These inequalities state that for each node i, the total capacity associated with the arcs leaving (respectively, entering) node i must be at least equal to the sum of demands for which node i is an origin (respectively, a destination).

In the case of the MCFND, BD will be compared to the same implementation of Benders decomposition except that a phase of local branching is performed after solving each master problem in the integer phase of the solution process. The local branching procedure uses as starting points the solutions to the master problems in the integer phase. This enhanced algorithm will be referred to in the following as BD-LB.

Because the local branching search can be performed many times, a limit (Time-LB) is imposed on the time spent each time this procedure is called. The optimizer used in the local branching phase is the Benders algorithm itself. An optimality gap of 1% ($\bar{\epsilon} = 1\%$) was used for the solution of the subproblems. Because the objective is to quickly identify several feasible solutions, the branching scheme is applied each time a new solution is identified whether the objective function value associated with this new solution is better than that of the previous one or not. The diversification procedure is applied in the case where the subproblem obtained is infeasible. As for parameter κ, it starts with value one for all runs of the local branching procedure.

Another remark to be made concerns the core point $\hat{x}^0$ that is used in the evaluation of the deepness of each cut: point $\hat{x}^0$ is fixed to $\hat{x}_{ij}^0 = 0.5 \ \forall (i,j) \in A$ at the beginning of the solution process and does not change. Two implementations of BD-LB will be tested. In BD-LB-All, all solutions found in the local branching phases will be used to generate cuts that will be added to the master problem, whereas in BD-LB-One, only the solution that is identified as generating the deepest cut will be added at each iteration. One should note that in both cases, cuts are added in addition to the one obtained for the solution to the master problem. Also, in both implementations, all neighbourhoods found to be infeasible are eliminated from the feasible region of the master problem using the local branching constraints.

All instances of the MCFND problem were obtained using the *mulgen* generator described in Crainic et al. (2001). A first set of instances are based on networks of 10 nodes, with 50, 60, 70, and 80 arcs, and 10 and 15 commodities. For each network size, five instances were generated randomly. This first set will be used to test the main algorithmic ideas proposed in this paper. A second set of instances, based on networks of 20 nodes, with 70 arcs, and 10 or 15 commodities will then be used to validate the results. In this case, only one instance was generated for each network size. For each network created, data were generated so as to obtain instances for which the relative difference between the fixed and variable costs

is either high compared with that of the arc capacities and demands, or low. Therefore, the first set of problems contains 80 instances, and the second set contains four instances.

All experiments were performed on a 2.4 GHz AMD Opteron 64-bit processor. A maximum time of 10,000 seconds was imposed on all runs for the first set of instances, and the optimality gap considered was $\epsilon = 1\%$. As for the parameter, Time-LB for both implementations using local branching, a first set of values was fixed to 1, 3, 5, and 7 seconds, respectively, for the instances based on networks of 50, 60, 70, and 80 arcs. These times were then divided and multiplied by two to obtain two more sets of values that will provide a better understanding of how the results are affected when the effort devoted to the local branching phases varies. It should be noted that BD-LB-All-j and BD-LB-One-j for $j = 1$, 2, and 3 refer to the runs for each implementation with Time-LB fixed to the smallest, base, and largest values, respectively.

To analyze the trade-offs in using local branching, one should first compare BD to BD-LB-All because BD-LB-All uses all information obtained by the local branching search. Therefore, the first set of problems is solved using BD and BD-LB-All when Time-LB is fixed to the base values (BD-LB-All-2). Comparing the results obtained, one may classify the instances as being easily solved (taking up less than 1,000 seconds of CPU), moderately hard to solve (between 1,000 and 10,000 seconds), and very hard to solve (more than 10,000 seconds) for the BD implementation. By using such a classification, one distinguishes 35 easy instances, 19 moderate instances, and 26 hard instances. Out of the 26 hard instances for BD, BD-LB-All-2 was able to solve 10 of them in less than 10,000 seconds. In total, BD-LB-All-2 is faster than BD for 70 out of the 80 instances. All instances for which BD-LB-All-2 is slower than BD are considered easy instances. For these 10 instances, the average solution time of BD is 71.26 seconds compared with 138.73 seconds for BD-LB-All-2. Therefore, on all instances solved by both implementations before the time limit is reached (a total of 54 instances), one observes that BD-LB-All-2 is either faster (for 44 instances) or a little slower (for 10 instances) than BD.

To further analyze these results, let us consider Table 1. The top of Table 1 reports the average, variance, and standard deviation (SD) for the solution times obtained by BD and BD-LB-All-2 on the easy and moderate instances. As for the hard instances, we report the average gap obtained on the 16 (out of the 26) instances that both BD and BD-LB-All-2 failed to solve in the maximum time allowed. One observes that on the easy instances, on average, BD-LD-All-2 is almost twice as fast as BD (84.21 seconds versus 163.68 seconds). On the moderate instances, the difference is even more significant because BD-LB-All-2 is more than four times faster, on average, than BD (696.82 seconds versus 3,021.13 seconds). For the hard instances, as was already noted, 10 out of the 26 instances were solved by BD-LB-All-2 before the time limit was reached. On some of these instances, the difference was quite large. For example, in one of these cases, BD-LB-All-2 took 1,239.2 seconds to complete the search, whereas BD was stopped with a gap of 9.14% after 10,000 seconds. When both BD and BD-LB-All-2 were unable to complete the search within the time limit, on average, BD obtained a gap of 10.66% compared with 5.71% obtained by BD-LB-All-2. This represents a relative gap reduction of 46.44%.

Table 1 BD vs. BD-LB-All-2

	BD			BD-LB-All-2		
Measures	Easy	Moderate	Hard	Easy	Moderate	Hard
Average	163.68 s	3,021.13 s	10.66%	84.21 s	696.82 s	5.71%
Variance	53,071	3,565,791	43	20,167	332,004	21
SD	230.37 s	1,888.33 s	6.52%	142.01 s	576.2 s	4.55%
Feas.	21.54	82.53	—	8.66	32.26	—
Opt.	65.91	272.47	—	37.14	131.53	—
No. It.	102.03	356.00	—	19.69	78.47	—

Another important observation concerns the standard deviations obtained by both implementations. For all three categories of instances, BD-LB-All-2 obtains smaller standard deviations than BD for the results considered in Table 1. In the case of the easy instances, one obtains a relative decrease of 38.36% in the standard deviation of the solution time when using BD-LB-All-2 compared to BD (142.01 seconds versus 230.37 seconds). For the moderate instances, the relative decrease is 69.49% (576.2 seconds versus 1,888.33 seconds). As for the hard instances, one observes a relative decrease of 30.21% in the standard deviation of the gap obtained (4.55% versus 6.52%). By adding local branching searches throughout the integer phase of the Benders solution process, one makes the resulting algorithm much more robust.

All these differences may be explained by analyzing the number of cuts added as well as the number of iterations executed by both implementations, which are reported at the bottom of Table 1. Feas., Opt., and No. It. refer, respectively, to the average number of feasibility and optimality cuts added, as well as the average number of iterations performed during the integer phase of the solution process. These results are not reported for the hard instances because the BD implementation failed to solve them. One should notice that for each iteration in the integer phase an integer master problem is solved. When considering the instances that BD is able to solve in the maximum time allowed, one observes that the total number

of cuts added by BD-LB-All-2 is, on average, much lower than BD. Because BD-LB-All-2 possibly adds multiple cuts at each iteration compared with only one in the case of BD, one can expect a reduction in the number of master problems solved in the integer phase when applying local branching. Indeed, when comparing the total number of iterations needed in the integer phase (No. It.) by both implementations, one can see that, on average, BD solves about five times more integer master problems in the case of the easy and moderate instances. In turn, this explains the increased total solution times for BD with respect to BD-LB-All-2.

The comparison between BD and BD-LB-All seems to demonstrate that with a relatively small effort spent in the local branching phases, one can obtain significantly better results when using Benders decomposition. This is particularly true for the cases where BD is less efficient. For moderate or hard instances, local branching can either considerably reduce the solution times or help to obtain better quality results.

One should now analyze how the results are affected when Time-LB varies, as well as the trade-off between using all feasible solutions to generate cuts (BD-LB-All) compared to using only the solution that is identified as generating the deepest cut (BD-LB-One). Table 2 reports the average results obtained by BD-LB-All-j and BD-LB-One-j for $j = 1, 2, 3$. Concerning the hard instances, one should mention that Hard–solv. represents those instances that were solved to optimality in the maximum time allowed by all runs of BD-LB-All and BD-LB-One. As for Hard–not solv., it represents those instances that were not solved by all runs of BD-LB-All and BD-LB-One. When considering all instances that were solved, both implementations obtain the best average results when Time-LB is fixed to the smallest values. As for those hard instances that were not solved completely, the difference in the average gaps is quite small. Therefore, it seems that one only needs to invest a limited effort in the local branching phases to obtain the best results when using either BD-LB-All or BD-LB-One. Also, results in Table 2 tend to show that whatever the values used for Time-LB or the strategy concerning the addition of the cuts, it seems always preferable to use local branching in the present case.

When directly comparing the average times obtained by BD-LB-All and BD-LB-One, one can

Table 2 BD-LB-All vs. BD-LB-One

	BD-LB-All			BD-LB-One		
Instances	1	2	3	1	2	3
Easy (s)	51.11	84.21	142.65	54.31	79.48	168.25
Moderate (s)	643.36	696.82	1,162.75	691.11	831.96	1,304.10
Hard–solv. (s)	2,788.45	3,662.10	4,784.63	3,011.77	3,139.48	3,822.59
Hard–not solv. (%)	6.61	6.20	6.64	6.27	6.83	6.68

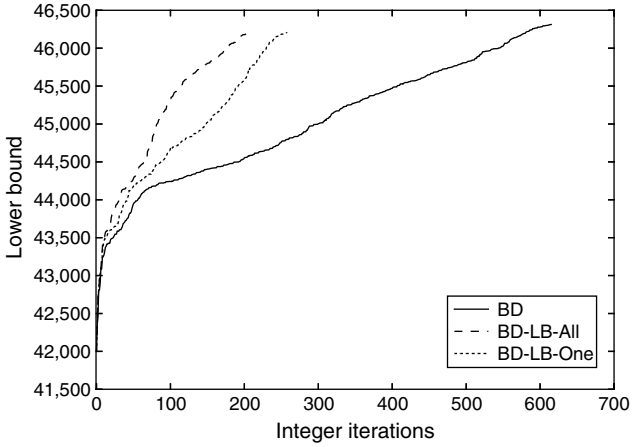

Figure 1 Integer Iterations vs. Lower Bound

observe a slight advantage in using BD-LB-All. To better understand the solution process of implementations BD, BD-LB-All-1, and BD-LB-One-1 in the case of a hard instance, let us consider Figures 1 and 2. Figure 1 shows how the lower bound increases through the iterations in the integer phase. One can clearly see that the use of local branching gives a steeper ascent to the lower bound. When considering the evolution of the upper bound, illustrated in Figure 2, one also observes a favorable decrease when local branching is used. For this particular instance, BD-LB-All-1 is faster than BD-LB-One-1 and the difference is due to the evolution of the lower bound. When using BD-LB-All, one seems to hedge against the risk of not using a necessary cut by adding all cuts identified. This aspect of the BD-LB-All implementation constitutes an advantage on certain hard instances. However, one should note that the average differences between BD-LB-All and BD-LB-One are relatively small. Therefore, criterion (19), which measures the depth of each cut, does seem to work fairly well.

To validate the observations made so far, a second set of tests was obtained on four instances based

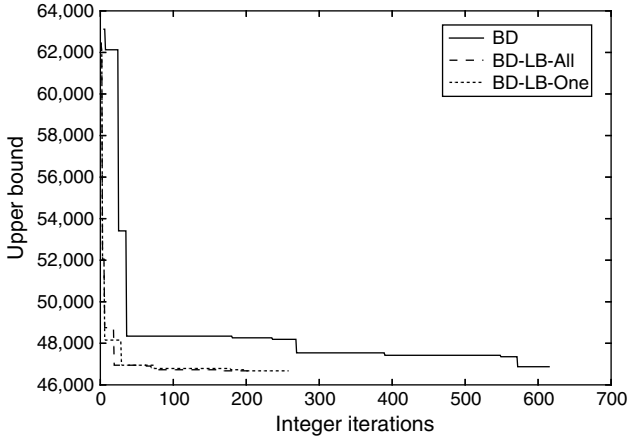

Figure 2 Integer Iterations vs. Upper Bound

on larger networks (20 nodes, 70 arcs, and 10 or 15 commodities). For this second set of tests, the maximum time allowed was increased to 30,000 seconds. Also, because the previous tests seemed to demonstrate that when using local branching in the Benders algorithm, one usually obtained the best results when Time-LB was fixed to the smallest values and was set to 2.5 seconds for all runs. Out of the four instances solved, BD only solved one in the maximum allotted time. For this instance, the solution times are 1,411.71 seconds for BD, 685.20 seconds for BD-LB-One, and 364.60 seconds for BD-LB-All. For the three remaining instances, one more problem is solved using the local branching strategies. In this case, BD finds a solution whose optimality gap is 6.97%, whereas BD-LB-One solves the problem in 25,177.90 seconds compared with 19,844.35 seconds for BD-LB-All. All implementations were unable to solve the two last instances for which the average gaps obtained are 9.84% for BD, 5.72% for BD-LB-One, and 6.83% for BD-LB-All. Once again, the use of local branching gives an added advantage when compared with BD.

4.2. Stochastic Integrated Model for Logistics Network Design

The model proposed by Cordeau et al. (2006) is a generalization of the MCFND that integrates several families of decisions that must be made when designing the logistics network of a manufacturing company in a single-country and single-period planning context. The formulation uses binary variables to represent location, product assignment, and sourcing decisions, and uses continuous variables to represent the flows of products in the network. Constraints impose capacity on production and distribution facilities as well as demand satisfaction and flow conservation for both finished products and raw materials.

The problem considered here is a stochastic version of the one originally considered by Cordeau et al. (2006). Because perfect information concerning the parameters of a model is rarely available, one can use stochastic programming to introduce uncertainty in optimization problems (Birge and Louveaux 1997). In our version of the problem, customer demands are considered to be stochastic. The logistics model can then be rewritten using the classical two-stage stochastic formulation with fixed recourse.

To define the stochastic model, let Ω be the set of all possible random events. Let us suppose that Ω is a discrete set. For $\omega \in \Omega$, let $p_\omega \in [0, 1]$ define the probability associated with event ω, where $\sum_{\omega\in\Omega} p_\omega = 1$. Now, let $\mathcal{F}$ be the set of finished products and let $\mathcal{R}$ denote the set of raw materials and purchased components used in the manufacturing or assembly of finished products. For every $r \in \mathcal{R}$ and every $f \in \mathcal{F}$, let b^{rf} be the quantity of raw material r required in the production of one unit of product f. The set of all suppliers is denoted by $\mathcal{S}$, and $\mathcal{S}^r \subseteq \mathcal{S}$ represents the subset of suppliers that can provide raw material $r \in \mathcal{R}$. Let also $\mathcal{P}$ and $\mathcal{W}$ denote the sets of actual and potential locations for plants and warehouses, respectively. For every product $f \in \mathcal{F}$, let $\mathcal{P}^f$ and $\mathcal{W}^f$ denote the subsets of plants and warehouses at which product f can both be made and stored. Finally, let $\mathcal{C}$ be the set of customer locations. For every $c \in \mathcal{C}$ and every $f \in \mathcal{F}$, let $a_c^f(\omega)$ be the demand of customer c for product f, which in this case depends on the observed random event $\omega \in \Omega$.

For notational convenience, denote by $\mathcal{K} = \mathcal{R} \cup \mathcal{F}$ the set of all commodities, and by $\mathcal{O} = \mathcal{S} \cup \mathcal{P} \cup \mathcal{W}$ and $\mathcal{D} = \mathcal{P} \cup \mathcal{W} \cup \mathcal{C}$ the sets of origins and destinations for these commodities, respectively. Then, for every $k \in \mathcal{K}$, define $\mathcal{O}^k \subseteq \mathcal{O}$ and $\mathcal{D}^k \subseteq \mathcal{D}$ as the sets of potential origins and destinations for commodity k, respectively. Associated with every $k \in \mathcal{K}$ and every $o \in \mathcal{O}^k$, let V_o^k be a binary variable, with cost c_o^k, taking the value one if and only if commodity k is assigned to origin o. Also, for every origin $o \in \mathcal{O}$, let U_o be a binary variable whose value is one if and only if this origin is assigned at least one commodity. Let c_o be the fixed cost of selecting origin $o \in \mathcal{O}$. It should be noted that the variables V_o^k and U_o define the first-stage decisions in the stochastic recourse model. These decisions must be made before demands become known. For every $k \in \mathcal{K}$ and $o \in \mathcal{O}^k$, let q_o^k be an upper limit on the amount of commodity k to be provided by origin o to any destination. Finally, for every $o \in \mathcal{O}$, let u_o be the capacity, in equivalent units, of origin o, and for every $k \in \mathcal{K}$, let u_o^k be the amount of capacity required by one unit of commodity k at origin o.

For every origin-destination pair $(o, d) \in \mathcal{O} \times \mathcal{D}$, let $\mathcal{M}_{od}$ be the set of transportation modes between o and d, and for every $k \in \mathcal{K}$, define $\mathcal{M}_{od}^k \subseteq \mathcal{M}_{od}$ as the subset of feasible transportation modes for commodity k. Now, for every $m \in \mathcal{M}_{od}^k$, define a nonnegative variable $X_{od}^{km}(\omega)$, with cost c_{od}^{km}, representing the number of units of commodity k transported from origin o to destination d using mode m if the observed random event is $\omega \in \Omega$. In the stochastic model, these variables define the second-stage decisions that are made once customer demands become known. These decisions represent the recourse that is available to the company given both the design decisions that were taken in the first stage and the random event that is observed. The model can now be stated as follows:

$$\min \sum_{o\in\mathcal{O}} c_o U_o + \sum_{k\in\mathcal{K}} \sum_{o\in\mathcal{O}^k} \left[c_o^k V_o^k + \sum_{\omega\in\Omega} p_\omega \left[\sum_{d\in\mathcal{D}^k} \sum_{m\in\mathcal{M}_{od}^k} c_{od}^{km} X_{od}^{km}(\omega) \right] \right] \quad (25)$$

$$\text{s.t.} \sum_{s\in\mathcal{S}^r}\sum_{m\in\mathcal{M}^r_{sp}} X^{rm}_{sp}(\omega) - \sum_{f\in\mathcal{F}^r}\sum_{w\in\mathcal{W}^f}\sum_{m\in\mathcal{M}^f_{pw}} b^{rf} X^{fm}_{pw}(\omega) = 0$$

$$\forall\, r\in\mathcal{R},\ \forall\, p\in\mathcal{P},\ \forall\,\omega\in\Omega, \quad (26)$$

$$\sum_{p\in\mathcal{P}^f}\sum_{m\in\mathcal{M}^f_{pw}} X^{fm}_{pw}(\omega) - \sum_{c\in\mathcal{C}^f}\sum_{m\in\mathcal{M}^f_{wc}} X^{fm}_{wc}(\omega) = 0$$

$$\forall\, f\in\mathcal{F},\ \forall\, w\in\mathcal{W}^f,\ \forall\,\omega\in\Omega, \quad (27)$$

$$\sum_{w\in\mathcal{W}^f}\sum_{m\in\mathcal{M}^f_{wc}} X^{fm}_{wc}(\omega) = a^f_c(\omega)$$

$$\forall\, f\in\mathcal{F},\ \forall\, c\in\mathcal{C}^f,\ \forall\,\omega\in\Omega, \quad (28)$$

$$\sum_{k\in\mathcal{K}}\sum_{d\in\mathcal{D}^k}\sum_{m\in\mathcal{M}^k_{od}} u^k_o X^{km}_{od}(\omega) - u_o U_o \le 0$$

$$\forall\, o\in\mathcal{O},\ \forall\,\omega\in\Omega, \quad (29)$$

$$\sum_{d\in\mathcal{D}^k}\sum_{m\in\mathcal{M}^k_{od}} X^{km}_{od}(\omega) - q^k_o V^k_o \le 0$$

$$\forall\, k\in\mathcal{K},\ \forall\, o\in\mathcal{O}^k,\ \forall\,\omega\in\Omega, \quad (30)$$

$$X^{km}_{od}(\omega) \ge 0$$

$$\forall k\in\mathcal{K}, \forall o\in\mathcal{O}^k, \forall d\in\mathcal{D}^k, \forall m\in\mathcal{M}^k_{od}, \forall\,\omega\in\Omega, \quad (31)$$

$$U_o\in\{0,1\} \quad \forall\, o\in\mathcal{O}, \quad (32)$$

$$V^k_o\in\{0,1\} \quad \forall\, k\in\mathcal{K}, \forall\, o\in\mathcal{O}^k. \quad (33)$$

The objective function (25) minimizes the sum of all fixed costs as well as the average variable costs over all possible random events. Considering all $\omega\in\Omega$, constraints (26) ensure that the total amount of raw material r shipped to plant p is equal to the total amount required by all products made at this plant, and constraints (27) ensure that all finished products that enter a given warehouse also leave that warehouse. Also, given all possible random events $\omega\in\Omega$, demand constraints are imposed by Equations (28). Finally, constraints (29) impose global capacity limits on suppliers, plants, and warehouses, whereas limits per commodity are enforced through (30). It should be noted that (29) and (30) are the linking constraints that bind both the first-stage and second-stage variables.

When considering problem (25)–(33), depending on set Ω, one may obtain either an infinite or, at least, a very large number of demand scenarios. If so, strategies such as the sample average approximation (SAA) approach can be used to quickly produce high-quality solutions (see, e.g., Santoso et al. 2005). The SAA approach generates a series of independent random samples of demand scenarios that are used to define approximation problems. As was shown in Santoso et al. (2005), each approximation problem can be solved efficiently using the L-shaped (or Benders) algorithm (Van Slyke and Wets 1969). Statistical lower and upper bounds are then obtained to estimate the optimality gap and produce a stopping criterion.

Because Benders decomposition is well suited for stochastic network design problems and because many probabilistic algorithms (such as SAA) repeatedly use it to solve approximation problems, the algorithmic approach that is proposed in this paper is further validated by performing tests on a series of stochastic approximation problems generated from the deterministic instances introduced by Cordeau et al. (2006). More precisely, we have considered the 12 basic instances of Cordeau et al. (2006). We then generated six cases for each of them by considering 20, 30, or 40 scenarios and by varying the interval in which random demands are generated. In the first case, the demand of a customer for a product follows a uniform distribution over the interval $[d/k, d]$, where d is the original demand in the deterministic instance and k is a parameter equal to 1.25. In the second case, this interval is $[d/k-1, d]$. This gave us a total of 72 test instances with varying degrees of difficulty.

All experiments were performed on a 2.4 GHz AMD Opteron 64-bit processor. A maximum time of 18,000 seconds was imposed on all runs for all instances, and the optimality gap considered was $\epsilon = 1\%$ for both the LP and integer phases.

4.2.1. BD Implementation. The Benders implementation that is used to solve the stochastic problem is the L-shaped version of the original algorithm of Cordeau et al. (2006). When Benders decomposition is applied to a stochastic approximation problem, one obtains a set of optimality cuts for each random scenario that is defined in the model. By using the L-shaped algorithm, whenever a feasible solution to the original problem is found, the optimality cuts associated with the scenarios can either be aggregated to produce only one cut, as was originally done in Van Slyke and Wets (1969), or each cut may be added to the relaxed master problem to produce a multicut strategy (Birge and Louveaux 1988). By adding an optimality cut for each scenario, the multicut strategy can produce a better approximation of the objective associated with the original problem. However, by using this approach, one makes the relaxed master problems both larger and harder to solve. In the present case, preliminary tests showed that the aggregated version was preferable to the latter. Therefore, only the aggregated version is used in our BD implementation.

The Benders algorithm developed in Cordeau et al. (2006) also included both the features proposed by McDaniel and Devine (1977) as well as Magnanti and Wong (1981). It was observed that the cuts generated during the LP phase provided a very good approximation of the feasible region of the original problem. In the results reported for the deterministic case, only a few integer master problems needed to be

solved to obtain an optimal solution. It should also be noted that the best runs of the algorithm included Pareto-optimal cuts for all optimality cuts added during both the LP and integer phases. Finally, a series of valid inequalities were added to the relaxed master problem at the beginning of the solution process to strengthen the relaxation. These constraints were useful to ensure a rapid convergence of the method. For a complete description of all valid inequalities added, we refer the reader to the paper of Cordeau et al. (2006). All these features are included in our new BD implementation.

4.2.2. BD-LB Implementation. To further improve the BD implementation, a specialized version of the local branching approach is proposed. To take advantage of the fact that cuts generated using the continuous relaxation of the master problem provide useful information, local branching is now applied in the LP phase. In doing so, the continuous solutions of the master problems (i.e., x^ν) are rounded, and the integer vector obtained (defined by $\bar{x}^\nu$) is used in the local branching scheme. It should be specified that the integrality constraints are also relaxed from the local branching subproblems explored. If $\Delta(x^\nu, \bar{x}^\nu) = \kappa^\nu$, then the local branching subproblem is defined using the following constraint: $\Delta(x, \bar{x}^\nu) \leq \lceil \kappa^\nu \rceil$. This particular approach is applied to define each subproblem explored during a local branching search phase. Once again, the branching decision is taken each time a new solution is identified to quickly obtain different cuts for the relaxed master problem. The diversification procedure is only applied in the case of infeasibility. Because the size of the neighbourhoods explored may be large and because local branching is applied after each master problem solved in the LP phase, a maximum of two subproblems is explored each time the local branching procedure is called.

Another important point concerns the problems that are used to create the local branching subproblems. In this specialized version, the original problem is only used to define the local branching subproblems for the first two iterations of the Benders algorithm. By proceeding in this way, it was found that large infeasible portions of set X can be eliminated using local branching constraints at an early stage of the solution process. However, because the original problem is quite large, this makes the local branching subproblems hard to solve. Therefore, once the initial iterations are passed, the local branching subproblems are defined by adding the neighbourhood constraints to the current relaxed master problem. In doing so, instead of finding optimal solutions to the subproblems of type (P_t), one finds solutions that define lower bounds. The cuts generated in this way may be weaker than the ones obtained by solving the local branching subproblems. Nevertheless, because the approximation defined by the relaxed master problem can be improved quickly by adding more cuts, the quality of the lower bounds obtained for the local branching subproblems is improved after each iteration. Toward this end, the All strategy is chosen for this BD-LB implementation. This choice was also motivated by the results obtained on the MCFND.

4.2.3. Computational Results. To properly analyze the results obtained by both implementations on the stochastic approximation problems, the 72 instances are divided in four categories. The Solvable category refers to all instances that are solved in the maximum allotted time by both implementations (total of 22 instances). The Hard–solv. category refers to all cases that are solved by the BD-LB implementation but for which the BD implementation is unable to solve in the maximum time allowed (total of 20 instances). The Hard–not solv. category groups all instances that neither BD nor BD-LB was able to solve but for which at least one feasible solution is found (total of 27 instances). Finally, the Undetermined category contains three instances that both implementations failed to solve and for which no feasible solution is obtained.

Table 3 includes the values obtained by both implementations for the lower bound, upper bound, gap, and solution time (in seconds). All results reported are averages over the number of instances in each category. One first observes that the Benders approach works quite well on the stochastic logistics model: the solutions obtained are, on average, near the optimal gap of $\epsilon = 1\%$. Overall, the LP phase is again essential in the solution process. It is observed that, on average, the first feasible solution obtained in the integer phase is very good (i.e., a gap between 1% and 3%). However, the relaxed masterproblem becomes extremely

Table 3 BD vs. BD-LB

		BD				BD-LB			
Categories	Freq.	Lower bound	Upper bound	Gap (%)	Time	Lower bound	Upper bound	Gap (%)	Time
Undetermined	3	7,670,018	—	—	—	7,699,303	—	—	—
Hard–not solv.	27	10,643,300	10,871,567	2.17	Max	10,674,907	10,870,243	1.86	Max
Hard–solv.	20	8,833,127	8,944,038	1.23	Max	8,868,793	8,939,988	≤1	163
Solvable	22	8,892,027	8,971,413	≤1	427	8,903,555	8,960,785	≤1	23

hard to solve when the integrality constraints are reintroduced. This proved to be a major difficulty for the BD implementation on 50 of the 72 instances tested (the Hard–solv., Hard–not solv., and Undetermined categories).

To compare BD and BD-LB, let us first consider the 22 instances that make up the Solvable category. On these instances, the average solution time of BD is 427 seconds compared with 23 seconds for BD-LB. The standard deviation is 1,626.65 seconds for BD versus 21.77 seconds for BD-LB. Therefore, for these instances, the BD-LB implementation is both much faster and more robust than BD. The difference is even more dramatic on the 20 instances of the Hard–solv. category. In this case, BD is unable to solve the instances in 18,000 seconds of computation time (the average gap obtained being 1.23%), whereas BD-LB solves all instances in an average time of 163 seconds. For this category, the local branching strategy proves to be invaluable to close the gap. For the Hard–not solv. category, both implementations fail to solve the instances in the maximum allotted time. However, the quality of the solutions found by BD-LB is slightly better than that obtained by BD (average gap of 1.86% compared with 2.17%). In all cases, the quality of both the upper and lower bounds is better when the local branching strategy is used. These results clearly confirm the previous observations made concerning the benefits of the approach proposed in this paper.

5. Conclusion

This paper proposes a novel way to accelerate the Benders decomposition algorithm by hybridizing it with local branching. When using the Benders algorithm, by performing a local branching search after each master problem that is solved, one can explore the neighborhood of the solution obtained for the master problem, which may or may not be feasible, to find different feasible solutions. By doing so, one can obtain better upper bounds at each iteration, and because each different feasible solution can be used to generate an optimality cut, one can also simultaneously obtain better lower bounds. Furthermore, the subproblems that are found to be infeasible during the local branching phases can be used as a way to complement or replace the feasibility cuts added during the solution process. Therefore, all information obtained through the local branching strategy can be exploited within the context of Benders decomposition.

Finally, because local branching only requires the use of a solver, if one can apply the Benders algorithm to the problem at hand, then the same Benders algorithm can be used as this optimizer. This makes the use of local branching possible for all 0-1 integer problems that can be solved by Benders decomposition. Numerical results reported in this paper have shown how local branching can greatly improve the performance of the Benders algorithm in the case of two integer linear problems. Recently, applications of Benders decomposition have focused on nonlinear or nonconvex optimization problems. Clearly, one interesting avenue of research would therefore be to apply our acceleration technique in those settings.

Acknowledgments

This research was partially funded by the Natural Sciences and Engineering Research Council of Canada under Grants 227837-04, 184121-05, and 38816-05. This support is gratefully acknowledged. The authors are also grateful to three anonymous referees for their valuable comments.

References

Benders, J. F. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* **4**(1) 238–252.

Birge, J. R., F. Louveaux. 1988. A multicut algorithm for two-stage stochastic linear programs. *Eur. J. Oper. Res.* **34**(3) 384–392.

Birge, J. R., F. Louveaux. 1997. *Introduction to Stochastic Programming*. Springer, New York.

Codato, G., M. Fischetti. 2006. Combinatorial Benders cuts for mixed-integer linear programming. *Oper. Res.* **54**(4) 756–766.

Cordeau, J.-F., F. Pasin, M. M. Solomon. 2006. An integrated model for logistics network design. *Ann. Oper. Res.* **144**(1) 59–82.

Costa, A. M. 2005. A survey on Benders decomposition applied to fixed-charge network design problems. *Comput. Oper. Res.* **32**(6) 1429–1450.

Côté, G., M. A. Laughton. 1984. Large-scale mixed integer programming: Benders-type heuristics. *Eur. J. Oper. Res.* **16**(3) 327–333.

Crainic, T. G., A. Frangioni, B. Gendron. 2001. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Appl. Math.* **112**(1–3) 73–99.

Fischetti, M., A. Lodi. 2003. Local branching. *Math. Programming* **98**(1-3) 23–47.

Geoffrion, A. M. 1970a. Elements of large-scale mathematical programming. Part 1: Concepts. *Management Sci.* **11**(11) 652–675.

Geoffrion, A. M. 1970b. Elements of large-scale mathematical programming. Part 2: Synthesis of algorithms and bibliography. *Management Sci.* **11**(11) 676–691.

Holmberg, K. 1990. On the convergence of cross decomposition. *Math. Programming* **47**(2) 269–296.

Holmberg, K. 1994. On using approximations of the Benders master problem. *Eur. J. Oper. Res.* **77**(1) 111–125.

Hooker, J. N. 2000. *Logic-Based Methods for Optimization*: *Combining Optimization and Constraint Satisfaction*. John Wiley & Sons, New York.

Magnanti, T. L., R. T. Wong. 1981. Accelerating Benders decomposition algorithmic enhancement and model selection criteria. *Oper. Res.* **29**(3) 464–484.

McDaniel, D., M. Devine. 1977. A modified Benders' partitioning algorithm for mixed integer programming. *Management Sci.* **24**(3) 312–319.

Santoso, T., S. Ahmed, M. Goetschalckx, A. Shapiro. 2005. A stochastic programming approach for supply chain network design under uncertainty. *Eur. J. Oper. Res.* **167** 96–115.

Van Roy, T. J. 1983. Cross decomposition for mixed integer programming. *Math. Programming* **25**(1) 46–63.

Van Slyke, R. M., R. J.-B. Wets. 1969. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM J. Appl. Math.* **17**(4) 638–663.