Computations with Disjunctive Cuts for Two-Stage Stochastic Mixed 0-1 Integer Programs

Lewis Ntaimo and Matthew W. Tanner
Department of Industrial and Systems Engineering, Texas A&M University, 3131 TAMU, College Station, TX 77843, USA, ntaimo@tamu.edu and mtanner@tamu.edu

Abstract

Two-stage stochastic mixed-integer programming (SMIP) problems with recourse are generally difficult to solve. This paper presents a first computational study of a disjunctive cutting plane method for stochastic mixed 0-1 programs that uses lift-and-project cuts based on the extensive form of the two-stage SMIP problem. An extension of the method based on where the data uncertainty appears in the problem is made, and it is shown how a valid inequality derived for one scenario can be made valid for other scenarios, potentially reducing solution time. Computational results amply demonstrate the effectiveness of disjunctive cuts in solving several large-scale problem instances from the literature. The results are compared to the computational results of disjunctive cuts based on the subproblem space of the formulation and it is shown that the two methods are equivalently effective on the test instances.

Keywords: stochastic programming; integer programming; disjunctive programming; lift-and-project cuts.

1 Introduction

Incorporating uncertainty in deterministic mixed-integer programming models leads to stochastic mixed-integer programming (SMIP) problems. Such models arise in many applications such as dynamic capacity acquisition (Ahmed and Garcia, 2003), supply chain planning (Alonso-Ayuso et al., 2003), vehicle routing (Laporte et al., 2002), and server location (Ntaimo and Sen, 2004). Stochastic programming allows for plans to be evaluated against possible future outcomes (scenarios) that represent alternative realizations of the problem data. In two-stage stochastic programming, one has to make first-stage decisions "here and now" without full information on a random event. Recourse actions are made in the second-stage after full information about the random event becomes available.

In this paper we focus on the following two-stage SMIP problem:

$$\begin{aligned} \text{SIP: Min } & c^\top x + E[f(x,\tilde{\omega})] \\ \text{s.t. } & x \in X \\ & x_i \in \{0,1\},\ \forall i \in I, \end{aligned} \tag{1}$$

where $x \in \Re^{n_1}$ is the first-stage decision vector, $c \in \Re^{n_1}$ is the first-stage cost vector, and $X = \{x \in \Re^{n_1}_+ \mid Ax \geq b\}$ with $b \in \Re^{m_1}$ as the right-hand side, and $A \in \Re^{m_1 \times n_1}$ as the

first-stage constraint matrix. $E[\cdot]$ is the mathematical expectation operator with respect to $\tilde{\omega}$, where

$$E[f(x,\tilde{\omega})] = \sum_{\omega\in\Omega} p_\omega f(x,\omega),$$

and $\tilde{\omega}$ is a multi-variate discrete random variable with an outcome (scenario) $\omega \in \Omega$ with probability of occurrence p_ω. For any scenario ω,

$$\begin{aligned} f(x,\omega) = \text{Min } & q(\omega)^\top y \\ \text{s.t. } & W(\omega)y \geq r(\omega) - T(\omega)x \\ & y \geq 0, y_j \in \{0,1\},\ \forall j \in J, \end{aligned} \tag{2}$$

where $y \in \Re^{n_2}$ is the recourse decision vector, $q(\omega) \in \Re^{n_2}$ is the cost vector, $r(\omega) \in \Re^{m_2}$ is the right-hand side, $T(\omega) \in \Re^{m_2 \times n_1}$ is the technology matrix, and $W(\omega) \in \Re^{m_2 \times n_2}$ is the recourse matrix. Throughout this paper, we assume that the constraints $-x \geq -1$ and $-y \geq -1$ are explicitly included in the constraint matrices A and $W(\omega)$, respectively. Problem (2) is generally referred to as the scenario problem or simply, the subproblem.

In this paper we consider instances of SIP (1-2) under the following assumptions:

(A1) Ω is a finite set.

(A2) $X = \{x \in \Re^{n_1}_+ \mid Ax \geq b\}$ is bounded by the hypercube defined by the constraints $0 \leq x \leq 1$.

(A3) For all $(x,\omega) \in X \cap \mathcal{B} \times \Omega$, $f(x,\omega) < \infty$.

Assumption (A3) is the relatively complete recourse (Wets, 1974) property, which requires that subproblem (2) remain feasible for all $(x,\omega) \in X \cap \mathcal{B} \times \Omega$.

Since we consider instances of SIP (1-2) under the assumption that Ω has finite support, we can rewrite the formulation (1-2) in extensive form (EF) as follows:

$$\begin{aligned} \text{EF: Min } & c^\top x + \sum_{\omega\in\Omega} p_\omega q(\omega)^\top y(\omega) \\ \text{s.t. } & Ax \geq b \\ & T(\omega)x\ + W(\omega)y(\omega) \geq r(\omega)\ \ \forall \omega \in \Omega \\ & x \geq 0, x_i \in \{0,1\},\ \forall i \in I \\ & y(\omega) \geq 0, y_j(\omega) \in \{0,1\},\ \forall j \in J,\ \forall \omega \in \Omega. \end{aligned} \tag{3}$$

Branch-and-cut type algorithms have seen great success in deterministic integer programming, benefiting from careful generation of cuts. However, it is only recently that such algorithms have had any success for SMIP. In fact, few comprehensive computational results have been reported for this class of problems. One recent approach for solving SMIP incorporates the theory of disjunctive programming (Balas, 1975, Blair and Jeroslow, 1978, Balas, 1979, Sherali and Shetty, 1980) within the context of stochastic programming. Disjunctive programming is used to generate valid inequalities (cuts) for the linear programming relaxation of the original problem.

The theory of disjunctive programming is first applied to SMIP in a dissertation by Carøe (1998) (also Carøe and Tind (1997)). The author proposes a scheme for solving a relaxation of mixed 0-1 SMIP problems. Using the dual block-angular structure of two-stage stochastic programs, the convex hull of feasible solutions for the EF relaxation is characterized using results from disjunctive programming. Carøe (1998) shows how a lift-and-project cut (Balas et al., 1993) can be generated for one subproblem and made valid for different outcomes. The cuts are generated in the $(x, y(\omega))$-space for each $\omega \in \Omega$.

Related to the decomposition method by Carøe (1998) is the disjunctive decomposition (D^2) method of Sen and Higle (2005). Unlike Carøe's method, the D^2 method uses cuts in a lower dimension $y(\omega)$-space of the scenario subproblem and draws on the "common-cuts-coefficients" theorem to generate a cut based on disjunctive programming for one scenario subproblem, and uses a simple translation to make it valid for different scenarios. However, this approach assumes fixed recourse (i.e. $W(\omega) = W$ for all $\omega \in \Omega$) and that $x \in vert(X)$. An extension of the D^2 method is the D^2-BAC method (Sen and Sherali, 2006) which allows for 'truncated' branch-and-bound in the second-stage. This method not only includes convexification of the second-stage feasible set, but also allows for the convexification of the second-stage value function based on the subproblem terminal nodal dual information from the branch-and-bound tree using a strategy from reverse convex programming (Sen and Sherali, 1987). More recently, Ntaimo (2006) has extended the D^2 approach to allow for random recourse. In this case, the cut coefficients are independently generated for each scenario subproblem.

The contributions of this paper include extending and implementing the decomposition method for SMIP proposed in Carøe (1998) and providing a computational experience with lift-and-project cuts for these problems. To the best of our knowledge, this is the first paper reporting on computations with this class of disjunctive cuts for SMIP. Also, we computationally compare the results obtained with Caroe's cuts with results obtained with the D^2 cuts of Sen and Higle (2005). We show that on our test instances, the two cuts provide comparable benefits in run time and so both types of cutting planes could be considered for future algorithms. We believe that the results presented in this paper will provide a motivation for new algorithms for SMIP based on disjunctive programming.

The rest of the paper is organized as follows. In the next section, we summarize some key theoretical results on lift-and-project cuts based on Carøe (1998) and make extensions. Using the theoretical results, in Section 3 we present variations of Caroe's algorithm for special cases of SMIP and highlight the connections with the D^2 method (Sen and Higle, 2005). We report on a computational experiment with the cutting plane approach on several large-scale instances from the literature in Section 4. We end the paper with some concluding remarks in Section 5.

2 Lift-and-Project Cuts for SMIP

We begin by reviewing the main theoretical results from disjunctive programming and lift-and-project cuts for SMIP based on Carøe (1998). Disjunctive programming deals

with linear programs with logical constraints. The results we present draw heavily upon the work of Balas et al. (1993) who derives lift-and-project cuts for deterministic IP. Even though the lift-and-project cuts are derived based on the LP relaxation of the EF (3), we will work with the two-stage LP relaxation of the original problem (1-2) given as follows:

$$\text{SLP: Min } c^\top x + E_{\tilde{\omega}}[f_c(x,\tilde{\omega})] \tag{4a}$$
$$x \in X, \tag{4b}$$

where for any realization ω of $\tilde{\omega}$,

$$f_c(x,\omega) = \text{Min } q(\omega)^\top y \tag{5a}$$
$$\text{s.t. } W(\omega)y \geq r(\omega) - T(\omega)x \tag{5b}$$
$$y \geq 0. \tag{5c}$$

Let us define the feasible set of the LP relaxation for a given ω based on constraining x and $y(\omega)$ as follows:

$$\begin{aligned}\mathcal{S}_{LP}(\omega) = \big\{ & x \in \Re^{n_1}_+, y(\omega) \in \Re^{n_2}_+ \mid Ax \geq b \\ & T(\omega)x + W(\omega)y(\omega) \geq r(\omega) \big\}\end{aligned}$$

Then the set of feasible solutions for ω becomes:

$$\begin{aligned}\mathcal{S}_{IP}(\omega) = \big\{ & (x, y(\omega)) \in \mathcal{S}_{LP}(\omega) \ \mid \\ & x_i \in \{0,1\}, \ \forall i \in I \\ & y_j(\omega) \in \{0,1\}, \ \forall j \in J \big\}\end{aligned}$$

Which means that the feasible set for the original problem can be written as

$$\mathcal{S} = \left\{ x, \{y(\omega)\}_{\omega\in\Omega} \mid (x, y(\omega)) \in \mathcal{S}_{IP}(\omega) \text{ for all } \omega \right\}.$$

Now let $(x^k, \{y^k(\omega)\}_{\omega\in\Omega})$ be a non-integer optimal solution to SLP (4-5) at some algorithmic iteration k. While Carøe (1998) focuses on eliminating non-integer solutions in the subproblems, in this paper we generalize the approach to allow for eliminating non-integer solutions in the first-stage as well. Let $z^k_\ell(\omega)$ denote a non-integer variable component of either x or $y(\omega)$ for some $\omega \in \Omega$ whose solution value is $\bar{z}^k_\ell(\omega)$. Then using $z^k_\ell(\omega)$ we can create the following two sets:

$$\mathcal{S}_{0,\ell}(\omega) = \big\{ x \in \Re^{n_1}_+, y \in \Re^{n_2}_+ \mid Ax \geq b \tag{6a}$$
$$T(\omega)x + W(\omega)y(\omega) \geq r(\omega) \tag{6b}$$
$$-z_\ell(\omega) \geq 0 \big\} \tag{6c}$$

and

$$\mathcal{S}_{1,\ell}(\omega) = \big\{ x \in \Re^{n_1}_+, y \in \Re^{n_2}_+ \mid Ax \geq b \tag{7a}$$
$$T(\omega)x + W(\omega)y(\omega) \geq r(\omega) \tag{7b}$$
$$z_\ell(\omega) \geq 1 \big\}. \tag{7c}$$

To eliminate the non-integer solution $(x^k, y^k(\omega))$, the following disjunction can be used:

$$\mathcal{S}(\omega) = \mathcal{S}_{0,\ell}(\omega) \cup \mathcal{S}_{1,\ell}(\omega). \tag{8}$$

The variable $z_\ell(\omega)$ is referred to as the *disjunction variable* in disjunctive decomposition (Sen and Higle, 2005). Notice that both (6) and (7) are non-empty due to assumptions (A1-A3), which ensure that the subproblems remain feasible for any restriction of the integer variables.

Now let $\lambda_{0,1}$ and $\lambda_{0,2}$ denote the vector of multipliers associated with (6a) and (6b), respectively, and let $\lambda_{0,3}$ denote the scalar multiplier associated with (6c). Let $\lambda_{1,1}$, $\lambda_{1,2}$ and $\lambda_{1,3}$ be similarly defined for (7a),(7b) and (7c), respectively. Also, define

$$I_i^k = \begin{cases} 1, & \text{if } i = \ell \\ 0, & \text{otherwise.} \end{cases}$$

and

$$I_j^k = \begin{cases} 1, & \text{if } j = \ell \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, the following LP can be used to generate a cut of the form $\gamma^\top(\omega)x + \pi^\top(\omega)y(\omega) \geq \nu(\omega)$ for scenario $\omega \in \Omega$:

$$\begin{aligned}
\text{Min} \quad & -\nu(\omega) + (x^k)^\top \gamma(\omega) + y^k(\omega)^\top \pi(\omega) \\
\text{s.t.} \quad & \gamma_i(\omega) - \lambda_{0,1}^\top A_i - \lambda_{0,2}^\top T_i(\omega) + I_i^k \lambda_{0,3} \geq 0, \quad \forall i \\
& \pi_j(\omega) - \lambda_{0,2}^\top W_j(\omega) + I_j^k \lambda_{0,3} \geq 0, \quad \forall j \\
& \gamma_i(\omega) - \lambda_{1,1}^\top A_i - \lambda_{1,2}^\top T_i(\omega) - I_i^k \lambda_{1,3} \geq 0, \quad \forall i \\
& \pi_j(\omega) - \lambda_{1,2}^\top W_j(\omega) - I_j^k \lambda_{1,3} \geq 0, \quad \forall j \\
& -\nu(\omega) + \lambda_{0,1}^\top b + \lambda_{0,2}^\top r(\omega) - \lambda_{0,3} \lfloor \bar{y}_{j(k)} \rfloor \geq 0 \\
& -\nu(\omega) + \lambda_{1,1}^\top b + \lambda_{1,2}^\top r(\omega) + \lambda_{1,3} \lceil \bar{y}_{j(k)} \rceil \geq 0 \\
& -1 \leq \gamma_i(\omega) \leq 1, \quad \forall i \\
& -1 \leq \pi_j(\omega) \leq 1, \quad \forall j \\
& -1 \leq \nu(\omega) \leq 1, \\
& \lambda_{0,1},\ \lambda_{0,2},\ \lambda_{0,3},\ \lambda_{1,1},\ \lambda_{1,2},\ \lambda_{1,3}, \geq 0.
\end{aligned} \tag{9}$$

A cut generated in this manner is referred to as a "lift-and-project" cut. LP (9) is formulated based on the *disjunctive cut principle* (Balas, 1975, Blair and Jeroslow, 1978) applied to the disjunction (8). In the formulation, A_i is the i-th column of A, $T_i(\omega)$ is the i-th column of $T(\omega)$, and $W_j(\omega)$ is the j-th column of $W(\omega)$. The objective of the LP is to maximize the (Euclidean) distance between the current point $(x^k, y^k(\omega))$ and the hyperplane $\gamma^\top(\omega)x + \pi^\top(\omega)y(\omega) \geq \nu(\omega)$. If the optimal objective value of (9) is non-negative it implies that the point $(x^k, y^k(\omega))$ is cut off.

The main drawback of lift-and-project cuts is that they are computationally expensive to generate, since they require solving an LP of size nearly double that of the system

defining either (6) or (7). Therefore, Carøe (1998) proposes generating a cut for one scenario, and then making it valid for other scenarios. This is done under additional assumptions on problem (1-2) and is summarized in the following three propositions. The first proposition is an extension based on the results in Carøe (1998), while the last two are restated from Carøe (1998) to fit our context.

PROPOSITION 2.1. *Let the recourse matrix $W(\omega)$ be random and technology matrix $T(\omega) = T$ and $r(\omega) = r$ for all $\omega \in \Omega$, and suppose that $\gamma^\top x + \pi^\top(\omega)y(\omega) \geq \nu$ is a cut obtained from* (9) *for some ω. Let the optimal solution to* (9) *be $(\pi(\omega), \gamma, \nu, \lambda_{0,1}, \lambda_{0,2}, \lambda_{0,3}, \lambda_{1,1}, \lambda_{1,2}, \lambda_{1,3})$. If for $\varpi \in \Omega, \varpi \neq \omega$ the problem*

$$\begin{aligned} \underset{\pi(\varpi)}{Min}\ & y^k(\omega)^\top \pi(\varpi) \\ s.t.\ \ & \pi_j(\varpi) \geq \lambda_{0,2}^\top W_j(\varpi) - I_j^k \lambda_{0,3}, \quad \forall j \\ & \pi_j(\varpi) \geq \lambda_{1,2}^\top W_j(\varpi) + I_j^k \lambda_{1,3}, \quad \forall j \\ & -1 \leq \pi_j(\varpi) \leq 1, \quad \forall j \end{aligned} \tag{10}$$

is feasible, then $\gamma^\top x + \pi^\top(\varpi)y(\varpi) \geq \nu$ is valid for $\mathcal{S}_{IP}(\varpi)$. If $\nu - \gamma^\top x^k - \pi^\top(\varpi)y^k(\varpi) > 0$, then $\gamma^\top x + \pi^\top(\varpi)y(\varpi) \geq \nu$ cuts off the point $(x^k, y^k(\varpi))$.

Proof. Follows the proof of Theorem 4 in Ntaimo (2006).

□

PROPOSITION 2.2. *(Carøe, 1998) Let the recourse matrix $W(\omega) = W$ and technology matrix $T(\omega) = T$ for all $\omega \in \Omega$, and suppose that $\gamma^\top x + \pi^\top y(\omega) \geq \nu(\omega)$ is a cut obtained from* (9)*. Then $\gamma^\top x + \pi^\top y(\varpi) \geq \nu(\varpi)$ is valid for $\mathcal{S}_{IP}(\varpi), \omega \neq \varpi, \varpi \in \Omega$, where*

$$\nu(\varpi) = \nu(\omega) + Min\ \{\lambda_{0,2}^\top(r(\varpi) - r(\omega)), \lambda_{1,2}^\top(r(\varpi) - r(\omega))\}, \tag{11}$$

and $\lambda_{0,2}$ and $\lambda_{1,2}$ are optimal solutions from (9). If $\nu(\varpi) - \gamma^\top x^k - \pi^\top y^k(\varpi) > 0$ then $\gamma^\top x + \pi^\top y(\varpi) \geq \nu(\varpi)$ cuts off the point $(x^k, y^k(\varpi))$.

Proof. See Proposition 3.1 in Carøe and Tind (1997).

□

PROPOSITION 2.3. *(Carøe, 1998) Let the recourse matrix $W(\omega) = W$, $\forall \omega \in \Omega$ and suppose that $\gamma^\top x + \pi^\top y(\omega) \geq \nu(\omega)$ is a cut obtained from* (9) *whose optimal solution is $(\pi, \gamma, \nu, \lambda_{0,1}, \lambda_{0,2}, \lambda_{0,3},$*

$\lambda_{1,1}, \lambda_{1,2}, \lambda_{1,3}$). *If for* $\varpi \in \Omega, \varpi \neq \omega$ *the problem*

$$
\begin{aligned}
\underset{\gamma(\varpi),\ \nu(\varpi),\ \lambda_{0,1},\ \lambda_{1,1}}{Min} \quad & (x^k)^\top \gamma(\varpi) - \nu(\varpi) \\
s.t. \quad & \gamma_i(\varpi) - \lambda_{0,1}^\top A_i \geq \lambda_{0,2}^\top T_i(\varpi) - I_i^k \lambda_{0,3}, \quad \forall i \\
& \gamma_i(\varpi) - \lambda_{1,1}^\top A_i \geq \lambda_{1,2}^\top T_i(\varpi) + I_i^k \lambda_{1,3}, \quad \forall i \\
& -\nu(\varpi) + \lambda_{0,1}^\top b \geq -\lambda_{0,2}^\top r(\omega) + \lambda_{0,3} \lfloor \bar{y}_{j(k)} \rfloor \\
& -\nu(\varpi) + \lambda_{1,1}^\top b \geq -\lambda_{1,2}^\top r(\varpi) - \lambda_{1,3} \lceil \bar{y}_{j(k)} \rceil \\
& -1 \leq \gamma_i(\varpi) \leq 1, \quad \forall i \\
& -1 \leq \nu(\varpi) \leq 1, \\
& \lambda_{0,1},\ \lambda_{1,1} \geq 0.
\end{aligned}
\tag{12}
$$

is feasible, then $\gamma^\top(\varpi)x + \pi^\top y(\varpi) \geq \nu(\varpi)$ *is valid for* $\mathcal{S}_{IP}(\varpi)$. *If* $\nu(\varpi) - \gamma^\top(\varpi)x^k - \pi^\top y^k(\varpi) > 0$, *then* $\gamma^\top(\varpi)x + \pi^\top y(\varpi) \geq \nu(\varpi)$ *cuts off the point* $(x^k, y^k(\varpi))$.

Proof. Follows the proof of Proposition 3.2 in Carøe and Tind (1997). □

REMARK 2.4. Proposition (2.3) is an extension of Proposition 5.3.3 in Carøe (1998) taking into account disjunctions based not only on y, but also on x. In the case of fixed recourse, both Propositions (2.2) and (2.3) allow a cut generated for one scenario to be made valid for the other scenarios by fixing the coefficients π and generating the coefficients $\gamma(\omega)$ and scalar $\nu(\omega)$ for each $\omega \in \Omega$. This idea is the essence of the Common-Cut-Coefficients theorem (Sen and Higle, 2005) in disjunctive decomposition, which allows for generating the π coefficients in the y-space while the right-hand coefficients $\gamma(\omega)$ and scalar $\nu(\omega)$ are determined for each $\omega \in \Omega$ via a simple translation. For SMIP with fixed recourse, the two propositions result in less proliferation of cuts in the subproblem as well as providing for a relatively less expensive way of generating cuts.

3 Disjunctive Cutting Plane Algorithms

We are now in a position to state a class of disjunctive cutting plane algorithms using the results from the previous section. We will refer to this class of algorithms as 'lift-and-project decomposition' (LPD) algorithms for SMIP, which are extensions of Carøe's algorithm (Carøe and Tind, 1997, Carøe, 1998). The extensions are twofold, (1) allowing the generation of lift-and-project cuts based on x and not only $y(\omega)$, and (2) allowing the generation of a cut based on one scenario and then making it valid for other scenarios for SMIP with random recourse but fixed technology matrix and right-hand side vector. Even though the algorithm in Carøe (1998) includes branch-and-bound based on branching on y, here we focus on a pure cutting plane approach.

Since the cutting plane approach involves sequential addition of cuts to the LP relaxation (4-5) of the original problem in order to refine the approximation of the convex hull of integer solutions, we redefine the problem data as follows. Let k denote

the algorithmic iteration index. Then for $k = 1$ we initialize the problem data with $T^1(\omega) = T(\omega)$, $W^1(\omega) = W(\omega)$, and $r^1(\omega) = r(\omega)$ for all $\omega \in \Omega$. At $k \geq 2$ the vector $\gamma^k(\omega)$ is appended to $T^{k-1}(\omega)$, $\pi^k(\omega)$ is appended to $W^{k-1}(\omega)$, and the scalar $\nu^k(\omega)$ is appended to $r^{k-1}(\omega)$. Following the L-shaped method (Slyke and Wets, 1969) for solving the LP relaxation, at algorithmic iteration k the problem has the following master program:

$$\begin{aligned} &\text{Min } c^\top x + \eta, \\ &\text{s.t. } Ax \geq b \\ &\qquad \beta_t^\top x + \eta \geq \alpha_t, t \in \Theta_k \\ &\qquad x \geq 0, \end{aligned} \tag{13}$$

where the second set of constraints are the optimality cuts with Θ_k being the index set for the L-shaped iterations at k. Note that feasibility cuts have been omitted due to assumption A3. The subproblem for $\omega \in \Omega$ takes the form:

$$\begin{aligned} f_c^k(x, \omega) = \text{Min } & q(\omega)^\top y \\ \text{s.t. } & W^k(\omega) y \geq r^k(\omega) - T^k(\omega) x \\ & y \geq 0. \end{aligned} \tag{14}$$

3.1 The Basic Lift-and-Project Decomposition (LPD) Algorithm

We can now state a basic LPD algorithm for SMIP as follows:

Basic LPD Algorithm:

Step 0. Initialization.
Set $k \leftarrow 1$, $U_1 \leftarrow \infty$, $L_1 \leftarrow -\infty$, $\epsilon > 0$, $T^1(\omega) \leftarrow T(\omega)$, $W^1(\omega) \leftarrow W(\omega)$, and $r^1(\omega) \leftarrow r(\omega)$, for all $\omega \in \Omega$, and let $x^0 \in X$ be given.

Step 1. Termination.
(a) If $U_k - L_k > \epsilon$ go to Step 2.
(b) If $U_k - L_k \leq \epsilon$ and an *incumbent* has been found, stop. Declare the *incumbent* optimal.
(c) If no *incumbent* has been found and x^k *stops changing*, impose integer restrictions on (13-14) and solve the problem to get an *incumbent* solution and update U_k. Compute $\epsilon' = U_k - L_k$, stop and declare the *incumbent* ϵ'-optimal.

Step 2. Solve LP Relaxation.
Solve (13-14) using the L-shaped method. Let $(x^k, \{y^k(\omega)\}_{\omega \in \Omega})$ be the LP optimal solution. If $(x^k, \{y^k(\omega)\}_{\omega \in \Omega})$ satisfy the integer restrictions, set $U_{k+1} \leftarrow \text{Min}\{c^\top x^k + E[f(x^k, \tilde{\omega})], U_k\}$, and if $c^\top x^k + E[f(x^k, \tilde{\omega})] < U_k$, store current solution as the *incumbent*, set $k \leftarrow k + 1$ and go to Step 1. Otherwise, set $L_{k+1} \leftarrow \text{Max}\{c^\top x^k + E[f_c(x^k, \tilde{\omega})], L_k\}$.

Step 3. Solve Cut Generation LPs and Perform Updates.
For $i \in I$ such that $0 < x_i < 1$ or $j \in J$ such that $0 < y_i(\omega) < 1$ choose a disjunction variable ℓ and form and solve (9) to obtain $(\gamma^k(\omega), \pi^k(\omega), \nu^k(\omega))$. Update $T^{k+1}(\omega) = [(T^k(\omega))^\top; \gamma^k(\omega)]^\top$, $W^{k+1}(\omega) = [(W^k(\omega))^\top; \pi^k(\omega)]^\top$ and $r^{k+1}(\omega) = [r^k(\omega),\ \nu^k(\omega)]$. Repeat this for all $\omega \in \Omega$. Set $k \leftarrow k+1$ and go to Step 1.

For practical size problems the pure cutting plane LPD algorithm may take an exponential number of iterations to converge to the optimal solution. Also, since we are not performing any branching, we may eventually get stuck with the *hull-relaxation* (Balas, 1984) of S as pointed out in Carøe (1998). This can be detected when x^k stops changing for several consecutive iterations in Step 1(c) of the algorithm. To terminate the algorithm we suggest computing an upper bound at this point which requires solving the subproblems to integer optimality. Computing the upper bound and setting $\epsilon' = U_k - L_k$ proves the incumbent solution is ϵ'-optimal.

3.2 Special Cases

The basic LPD algorithm can be adapted for SMIP with further properties on the problem data as outlined in Propositions 2.1, 2.2 and 2.3. To accomplish this we need to modify the cut generation Step 3 of the basic LPD algorithm. Next we state three algorithms for the following three special cases for SMIP with: (1) random recourse, fixed technology matrix, and fixed right-hand sides, (2) fixed recourse, fixed technology matrix, and random right-hand sides, (3) fixed recourse, random technology matrix, and random right-hand sides, and (4) pure binary first stage.

LPD**-1 Algorithm:** Random $W(\omega)$, fixed T and fixed r.

Step 3. Solve Cut Generation LPs and Perform Updates.
(a) Choose a scenario $\omega \in \Omega$ and a disjunction variable ℓ for which $0 < x_i^k < 1,\ i \in I$ or $0 < y_i^k(\omega) < 1, j \in J$. Form and solve (9) to obtain $(\gamma^k(\omega), \pi^k(\omega), \nu^k(\omega))$. Update $T^{k+1} = [(T^k)^\top; \gamma^k]^\top, W^{k+1}(\omega) = [(W^k(\omega))^\top; \pi^k(\omega)]^\top$ and $r^{k+1} = [r^k,\ \nu^k]$.
(b) For all $\varpi \neq \omega, \varpi \in \Omega$ such that $0 < x_\ell^k < 1,\ i \in I$ or $0 < y_\ell^k(\varpi) < 1, j \in J$ use the solution from Step 3(a) to form and solve (10) to get $\pi^k(\varpi)$. If problem is feasible update $W^{k+1}(\varpi) = [(W^k(\varpi))^\top; \pi^k(\varpi)]^\top$. Set $k \leftarrow k+1$ and go to Step 1.

LPD**-2 Algorithm:** Fixed W, fixed T and random $r(\omega)$.

Step 3. Solve Cut Generation LPs and Perform Updates.

(a) Choose a scenario $\omega \in \Omega$ and a disjunction variable ℓ for which $0 < x_i^k < 1,\ i \in I$ or $0 < y_i^k(\omega) < 1, j \in J$. Form and solve (9) to obtain $(\gamma^k, \pi^k, \nu^k(\omega))$. Update $T^{k+1} = [(T^k)^\top; \gamma^k(\omega)]^\top, W^{k+1} = [(W^k)^\top; \pi^k]^\top$ and $r^{k+1}(\omega) = [r^k(\omega),\ \nu^k(\omega)]$.
(b) For all $\varpi \in \Omega, \varpi \neq \omega$ such that $0 < x_\ell^k < 1,\ i \in I$ or $0 < y_\ell^k(\varpi) < 1, j \in J$ use the solution from Step 3(a) to computer $\nu^k(\varpi)$ using (11) and update $r^{k+1}(\varpi) = [r^k(\varpi),\ \nu^k(\varpi)]$. Set $k \leftarrow k + 1$ and go to Step 1.

LPD-**3 Algorithm:** Fixed W, random $T(\omega)$ and random $r(\omega)$.

Step 3. Solve Cut Generation LPs and Perform Updates.
(a) Choose a scenario $\omega \in \Omega$ and a disjunction variable ℓ for which $0 < x_i^k < 1,\ i \in I$ or $0 < y_i^k(\omega) < 1, j \in J$. Form and solve (9) to obtain $(\gamma^k(\omega), \pi^k, \nu^k(\omega))$. Update $T^{k+1}(\omega) = [(T^k(\omega))^\top; \gamma^k(\omega)]^\top, W^{k+1} = [(W^k)^\top; \pi^k]^\top$ and $r^{k+1}(\omega) = [r^k(\omega),\ \nu^k(\omega)]$.
(b) For all $\varpi \neq \omega, \varpi \in \Omega$ such that $0 < x_\ell^k < 1,\ i \in I$ or $0 < y_\ell^k(\varpi) < 1, j \in J$ use the solution from Step 3(a) to form and solve (12) to get $(\gamma^k(\varpi), \nu^k(\varpi))$. Update $T^{k+1}(\varpi) = [(T^k(\varpi))^\top; \gamma^k(\varpi)]^\top$, and $r^{k+1}(\varpi) = [r^k(\varpi),\ \nu^k(\varpi)]$. Set $k \leftarrow k + 1$ and go to Step 1.

Modification for SMIP with Pure Binary First Stage

For the case where the first-stage has pure binary variables, Step 1 and Step 2 of the algorithms can be modified as follows in order to guarantee finite convergence. In Step 2, solve the problem using the L-shaped method, but enforce the binary restrictions on the first-stage variables in the master program. In part (c) of Step 1, instead of terminating the algorithm, solve the subproblems as IPs and generate an optimality cut according to the algorithm of Laporte and Louveaux (1993) for SMIP, which we will refer to as the 'L^2 algorithm'. Add the optimality cut to the master program and continue to Step 2. This modification allows for convergence of the algorithm to an optimal solution as a direct consequence of the convergence of the L^2 algorithm.

3.3 Improved Cut Generation

A variety of computational "tricks" can be used to improve the run time of this algorithm. These include procedures for starting and stopping generation of cuts, different normalizations for the cut generation LP, rounding to strengthen the cuts, and lifting cuts generated by a smaller LP, possibly in the context of a branch-and-bound framework. The last two of these methods are described in Balas et al. (1993, 1996).

In our description of the basic *LPD* algorithm, we only generated one round of cuts per iteration. An alternative procedure could be to start adding cuts once the L-shaped

algorithm has terminated and then adding a preset number n rounds of cuts before restarting the L-shaped algorithm to find the next relaxed solution. Also, since finding the optimal master solution is the true goal of this algorithm, cuts could be added until the solution to the master program stops changing as a result of the cuts.

In (9), we use the normalization $-1 \leq \gamma_i(\omega) \leq 1,\ \forall i$ and $-1 \leq \pi_j(\omega) \leq 1,\ \forall j$ in order to ensure that we find a feasible solution and hence a valid inequality. However, there are other possible normalizations that provide tighter cuts. In the case of non-zero righthand sides, the normalization $\nu(\omega) = 1$ or $\nu(\omega) = -1$ can be used. The advantages to this normalization is that the cut generating linear program can be shrunk because the $\gamma(\omega)$ and $\pi(\omega)$ variables can also be fixed and it is sometimes possible to determine whether it is better to fix the value of $\nu(\omega)$ to 1 or to -1 (see Balas et al. (1993)). However, this normalization loses the guarantee that the program will have a finite optimal value. Another possible normalization is to make the sums of the absolute values of the $\gamma(\omega)$ and $\pi(\omega)$ equal to 1. As with our original normalization, this normalization also guarantees a finite solution to (9).

After a round of cuts has been generated, it is possible to strengthen the cuts through a rounding procedure. The basic idea of this procedure is to use the integrality conditions on the variables other than the disjunction variable to find some tighter parameter values for the cut. The full description of this tightening procedure can be found in Balas et al. (1993, 1996).

It is also possible to solve a variation of formulation (9) on some restriction of the variables. Then the cut generated from that restricted space can be lifted to the entire space. The main purpose of this technique is to allow for cuts generated at individual nodes of a branch-and-bound tree to be made valid for the entire tree, but it can also be used in the context of our algorithm by choosing proper restriction of variables (perhaps the non-integer variables of a given solution). The main benefit from doing this is to solve a smaller linear program than might otherwise have to be solved. Again, a full description of the lifting procedure can be found in Balas et al. (1993, 1996).

3.4 Comparison with the D^2 Method

Some comments on the differences between the D^2 method (Sen and Higle, 2005) and the basic LPD method are now in order. While the D^2 method is designed specifically for SMIP with fixed recourse having $x \in vert(X)$, the LPD method is applicable to a wider class of SMIP problems. Convergence in both methods is only guaranteed for mixed-binary second-stage. The difference between the two methods from a cut generation point of view is in how the disjunctive cuts are generated. The D^2 algorithm generates D^2-cuts which are in the subproblem $y(\omega)$-space while the LPD lift-and-project cuts are in the $(x, y(\omega))$-space of the extensive form (EF) (3).

From an algorithmic perspective, the D^2 method seeks to iteratively generate the convex hull of the subproblems for every x^k. In this case the cut is guaranteed to be a facet for the subproblem LP relaxation at that specific right-hand side value. The advantage of the D^2 cuts is that the lower dimension allows more flexibility in finding

cuts that will be useful when the x solution changes. Even though there is no guarantee of quality for the cuts once the x solution changes, computational experience seems to show that early cuts do help in closing the lower and upper bound gap (Ntaimo and Sen, 2004). In the LPD method the cuts are generated based on the solution of the LP relaxation of the EF. Thus these cuts are facets of the problem restricted to the larger dimension $(x, y(\omega))$-space.

4 Computational Results

We designed some computational experiments to gain insight into the effectiveness of the disjunctive cuts for closing the LP relaxation optimality gap of large-scale stochastic mixed-binary programs. Specifically, we apply the LPD algorithm to large-scale instances from two problem classes from the literature, namely, strategic supply chain (SSCh) planning under uncertainty (Alonso-Ayuso et al., 2003), and stochastic server location problems (SSLPs) (Ntaimo and Sen, 2004). We wanted to study the problem characteristics for which the LPD cuts might be particularly effective. We compare our results to those obtained from the D^2 algorithm, which has previously been shown to be successful on the two problem classes (Ntaimo and Sen, 2006). We also report on the performance of the ILOG CPLEX 9.0 MIP solver (ILOG, 2003) applied directly to the extensive form (EF) of the SSLPs.

In our experiments, the D^2 algorithm is implemented as follows. First, the L-shaped algorithm is used to solve the LP relaxation of the problem with the binary restrictions enforced on the first-stage variables in the master program. Second, disjunctive cuts are added until the first-stage decision stops changing after re-solving the relaxation. Then the subproblems are solved to integer optimality and an L^2 cut generated and added to master program. Finally, the algorithm returns to the second step unless the optimal solution has been found. We conducted all our experiments on an Optiplex GX620 with a Pentium D processor running at 3.0Hz with 3.5GB RAM. The problem instances were run to optimality or stopped when a CPU time limit of 10,800 seconds (3hrs) was reached.

4.1 Stochastic Supply Chain Planning Problems

We tested the LPD algorithm on a set of very large stochastic mixed integer program instances arising in strategic supply chain (SSCh) planning under uncertainty generated by Alonso-Ayuso et al. (2003). The objective function of the SSCh is to maximize the profit from expanding and running a given supply chain. The first-stage decision variables are pure binary and are for initial strategic decisions of plant capacity. The second-stage decision variables are mixed binary. The binary decisions are strategic recourse decisions about plant capacity, which products are processed, and which products are purchased from which vendors. The continuous decisions are for operating the supply chain. The uncertainty appears in the net profit parameter in the second-stage objective function, and in the demand parameter in the right-hand side of the second-stage problems. A version of the problem formulation is given in the Appendix.

Table 1 gives the problem characteristics for these instances. The headings of the table are as follows: 'Constrs' is the number of constraints, 'Bins' is the number of binary decision variables, 'Cvars' is the number of continuous decision variables, and 'Tot. Vars' is the total number of variables in the instance. The extensive form (EF) of the instances were too large for CPLEX to solve so we were unable to use that as a benchmark for our results. However, since the first-stage decisions are pure binary, we are able to compare these results with the results from the D^2 algorithm. Since the uncertainty in these problems appears both in the objective value and the right-hand side, we used the LPD-2 case of the algorithm to solve the problems.

Table 1: SSCh Instance Dimensions.

	EF				First-Stage		Second-Stage			
Case	Constrs	Bins	Cvars	Tot. Vars	Constrs	Bins	Constrs	Bins	Cvars	Tot. Vars
c1	76,318	899	67,551	68,450	73	71	3315	36	2,937	2,973
c2	77,928	900	70,564	71,464	73	72	3385	36	3,068	3,104
c3	70,795	895	61,249	62,144	70	67	3075	36	2,663	2,699
c4	76,775	897	70,495	71,392	70	69	3335	36	3,065	3,101
c5	88,743	906	84,042	84,948	78	78	3855	36	3,654	3,690
c7	69,411	895	58,489	59,384	66	67	3015	36	2,543	2,579
c8	87,824	906	83,582	84,488	79	78	3815	36	3,634	3,670
c10	69,871	895	58,259	59,154	66	67	3035	36	2,533	2,569

Table 2 shows the computational results for the LPD algorithm. The columns of this table given in order are, the number of iterations of the algorithm, the number of LPD cuts that were added, the number of L^2 cuts that were added, the CPU time taken, the lower and upper bounds given by the algorithm, and finally, the best solution found by Alonso-Ayuso et al. (2003). The main characteristic of these results is that within our time limit, the algorithm is able to obtain comparable and often improved solutions to those obtained by Alonso-Ayuso et al. (2003) with their branch-and-fix coordination (BFC) algorithm. In most of the cases, the algorithm either found the optimal solution, or was able to find relatively tighter upper and lower bounds on the optimal solution. This shows the effectiveness of the lift-and-project cuts towards solving this class of problems, and demonstrate that the LPD method has promise as a technique for solving stochastic integer programs of a practical size.

Ntaimo and Sen (2006) show that the D^2 is quite effective on these problems and so we ran our implementation of the D^2 algorithm on the instances for comparison. Table 3 gives the results of the D^2 algorithm. An interesting observation about Tables 2 and 3 is that the results of the algorithms are very similar. In every case but one, both algorithms found the same upper bounds, and always found the same lower bounds. This gives strong evidence that the D^2 cuts and the lift-and-project cuts are equally effective for this class of problems.

Figure 1 shows the characteristic plot of the convergence of lower and upper bounds for the two algorithms on the SSCh test set over time. The most important aspect of these plots is that both algorithms are able to converge to a near optimal solution in a short amount of time. In general, the convergence of the lower and upper bounds for the

Table 2: LPD Computational Results for SSCh.

Instance	Iters	LPD Cuts	L^2 Cuts	CPU (secs)	Z_{LB}	Z_{UB}	Z_{IP} BFC
c1	296	1127	41	10932.50	-206760	-184439	-178366.79
c2	68	69	0	80.22	0	0	0.00*
c3	529	3243	131	10847.60	-249903	-230268	-224564.20
c4	466	2576	103	10876.50	-222740	-201454	-197487.36
c5	320	713	18	10881.60	-278.49	724.783	0.00*
c7	772	3197	115	10834.30	-151205	-137217	-144181.28*
c8	344	1449	38	10915.40	-119767	-100523	-89607.39
c10	460	2645	102	10848.00	-153033	-139739	-139738.36*

LPD algorithm stopping tolerance used is $UB - LB < 0.0001$ or 3 hrs.
* Optimality has been proven by Alonso-Ayuso et al. (2003).

Table 3: D^2 Computational Results for SSCh.

Instance	Iters	D^2 Cuts	L^2 Cuts	CPU (secs)	Z_{LB}	Z_{UB}	Z_{IP} BFC
c1	270	3772	69	10876.80	-206760	-184439	-178366.79
c2	57	299	1	96.18	0	0	0.00*
c3	288	4945	151	10870.70	-249903	-230268	-224564.20
c4	225	4094	70	10801.50	-222740	-201454	-197487.36
c5	237	1633	33	10836.90	-278.49	0	0.00*
c7	295	5037	142	10909.80	-151205	-137217	-144181.28*
c8	254	3036	79	10963.90	-119767	-100523	-89607.39
c10	310	5267	160	10836.50	-153033	-139739	-139738.36*

D^2 algorithm stopping tolerance used is $UB - LB < 0.0001$ or 3 hrs.
* Optimality has been proven by Alonso-Ayuso et al. (2003).

two algorithms is similar. In the case of c5, the D^2 algorithm closes the gap much earlier than the LPD algorithm. This observation is backed up by the similarity of the solution times and optimality gaps for all the other instance runs. Since these algorithms tend to converge to a pretty good solution quickly and then stay stable for the rest of the 3 hours, one can terminate the algorithm once it has found that initial stable first-stage solution.

4.2 Stochastic Server Location Problems

We also tested the LPD method on randomly generated two-stage stochastic combinatorial optimization problem instances arising in server location under uncertainty (Ntaimo and Sen, 2004). The test instances we use are reported in Ntaimo and Sen (2006) and involve random instances each with five replications to guard against pathological cases. In these stochastic server location problems or SSLPs, the first-stage decision variables are binary and represent the potential "server" locations and take a value of 1 if a server is located at a given site, and 0 otherwise. The second-stage decision variables are mixed-binary with the binary variables taking a value 1 if a given "client" i is served by a server at given site, and 0 otherwise. The continuous variables represent overflow or unmet resource demand. Only one server can be located at any site and each client is served by

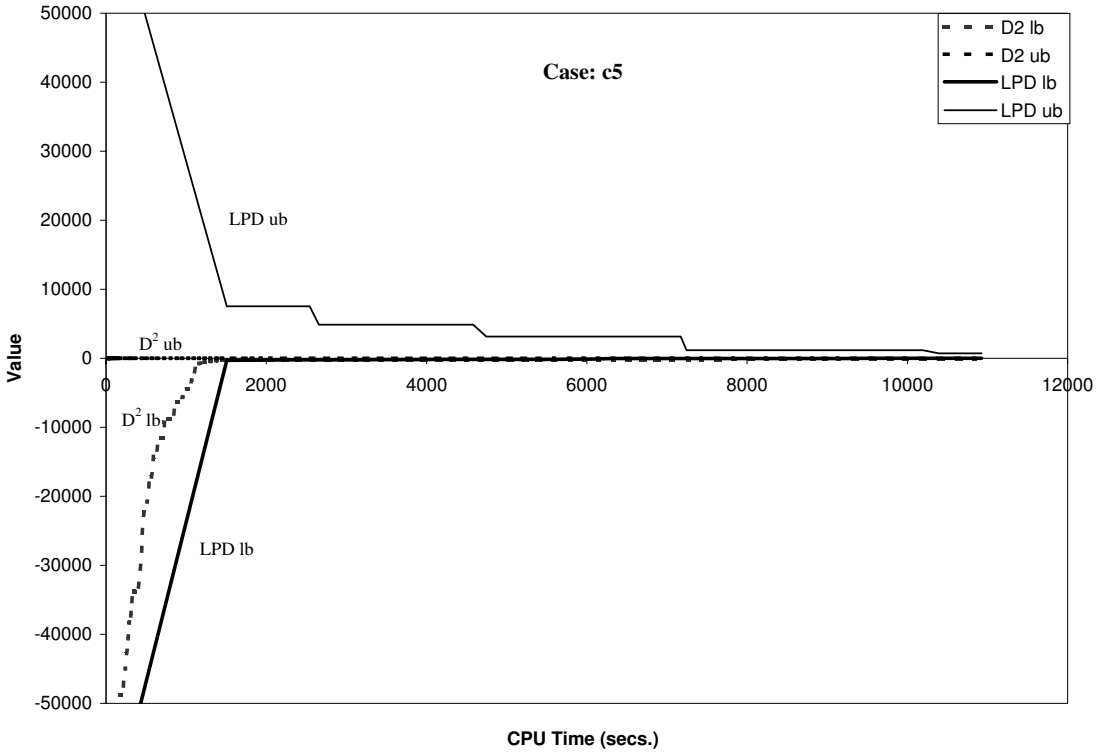

Figure 1: Convergence of lower and upper bounds for c5

only one server. The randomness in the model appears only in the right-hand side and is defined by whether or not a client is available to demand service. The SSLP formulation and a short description of the model are given in the Appendix.

We chose to test the LPD algorithm on these instances because they are a large set of SMIP test problems with published computational results using the D^2 method. These instances allow us to test the performance of the LPD algorithm when varying the number of scenarios, first-stage decision variables, and second-stage decision variables. This gives us insight into the strengths and weaknesses of the LPD cuts for different problems of different sizes. Since all of the randomness in these problems appears in the right-hand side only, the LPD-2 case of the algorithm is used in our experiments. Also, since the master variables are pure binary, the modification given at the end of section 3 is used to guarantee finite convergence of the algorithm.

Table 4 gives the problem characteristics. The problem instances are named according to the convention SSLP$m.n.S$, where m is the number of potential server locations, n is the number of potential clients, and $S = |\Omega|$ is the number of scenarios.

The computational results from the LPD algorithm are given in Table 5. The column heading 'L-shaped Iters' refers to the total number of iterations of the L-shaped algorithm during solution of the problem relaxation and 'LPD Cuts' is the number of lift-and-project cuts generated. Translated cuts are counted as distinct cuts, for example, 2000 cuts for problem instance SSLP10.50.2000 means that one disjunctive cut was generated,

Table 4: SSLP Instance Dimensions.

	EF				Second-Stage			
Instance	Cons	Bins	Cvars	Tot. Vars	Cons	Bins	Cvars	Tot. Vars
SSLP5.25.50	1,501	6,255	250	6,505	30	130	5	135
SSLP5.25.100	3,001	12,505	500	13,005	30	130	5	135
SSLP5.50.50	2,751	12,505	250	12,755	55	255	5	260
SSLP5.50.100	5,501	25,005	500	25,505	55	255	5	260
SSLP10.50.50	3,001	25,010	500	25,510	60	510	10	520
SSLP10.50.100	6,001	50,010	1,000	51,010	60	510	10	520
SSLP10.50.500	30,001	250,010	5,000	255,010	60	510	10	520
SSLP10.50.1000	60,001	500,010	10,000	510,010	60	510	10	520
SSLP10.50.2000	120,001	1,000,010	20,000	1,020,010	60	510	10	520
SSLP15.45.5	301	3,390	75	3,465	60	690	15	705
SSLP15.45.10	601	6,765	150	6,915	60	690	15	705
SSLP15.45.15	901	10,140	225	10,365	60	690	15	705
SSLP15.45.20	1201	13,515	300	13,815	60	690	15	705
SSLP15.45.25	1501	16,890	375	17,265	60	690	15	705

and then translated for the rest of the scenarios. The column 'L^2Cuts' refers to the number of times the subproblems were solved to integer optimality and an optimality cut of Laporte and Louveaux (1993) was generated and added to the master program. The column 'LPD CPU' gives the minimum, maximum, and average times, respectively, for each of the instances.

As benchmarks, we also applied the CPLEX MIP solver directly on the EF instances. The last column of Table 5 gives the CPU time and % gap reported by CPLEX. The results show that CPLEX is unable to solve the larger instances to optimality within the time limit, and the % gap is generally larger as the number of scenarios increases. To solve the EF instances, we used the CPLEX MIP solver with the following parameter settings suggested in Ntaimo and Sen (2004): "set mip emphasis 1" (emphasizes looking for feasible solutions), "set mip strategy start 4" (uses barrier at the root), and "branching priority order on x" (first branches on any fractional component of x before branching on y). Based on previous experimentation, these parameters seemed to give the best CPU time for these instances.

The results show that the LPD algorithm is quite successful for solving these particular instances. The worst time for any of the problems is slightly over 30 minutes and every instance was solved to optimality. The most striking aspect of the results is that only a few lift-and-project cuts are needed to solve the large instances. Most tests only needed one round of cuts before converging to the optimal solution. Instance SSLP15.45.25 has the largest average number of cuts, needing up to four sets. This shows the ability of translated lift-and-project cuts to improve algorithm performance without too much computational effort. Like other decomposition algorithms, the LPD algorithm seems to be relatively insensitive to the number of scenarios, but highly sensitive to the number of first-stage decision variables.

Ntaimo and Sen (2006) solved the SSLP instances using the D^2 algorithm. Using our implementation of the algorithm, we obtained the results given in Table 6 for comparison.

Table 5: LPD Computational Results for SSLP with Five Replications.

	Iters	LPD Cuts	L^2 Cuts	CPU (secs)			CPLEX on EF	
Problem	Avg	Avg	Avg	Min	Max	Avg	CPU	Gap
SSLP5.25.50	18.67	33.33	0.50	0.12	0.20	0.17	1.73	0.00%
SSLP5.25.100	18.83	83.33	0.67	0.14	0.30	0.23	130.20	0.00%
SSLP5.50.50	17.60	0.00	0.00	0.11	0.16	0.15	1.30	0.00%
SSLP5.50.100	18.00	0.00	0.00	0.20	0.27	0.24	3.61	0.00%
SSLP10.50.50	250.50	50.00	1.83	16.77	48.60	29.36	>10800	0.30%
SSLP10.50.100	250.83	100.00	2.00	21.00	57.17	32.97	>10800	0.74%
SSLP10.50.500	268.83	500.00	2.00	70.77	114.57	99.56	>10800	12.57
SSLP10.50.1000	269.83	1000.00	2.00	124.57	183.42	160.88	>10800	24.17
SSLP10.50.2000	275.33	2000.00	1.83	139.38	344.69	264.47	>10800	39.74
SSLP15.45.5	283.50	19.17	4.33	10.36	1579.51	311.81	>10800	0.67%
SSLP15.45.10	610.00	23.33	2.83	38.52	1421.21	599.53	>10800	0.52%
SSLP15.45.15	429.50	47.50	3.67	26.89	2017.18	564.23	>10800	1.50%
SSLP15.45.20	315.40	44.00	2.80	57.06	1827.37	505.40	>10800	1.00%
SSLP15.45.25	381.20	95.00	4.00	259.76	943.96	511.63	>10800	0.73%

LPD algorithm stopping tolerance used is 0.0001% Gap.

Like with the SSCh instances, the results in Tables 2 and 3 are very similar, probably and indication of the similarity of the strength of the cuts generated by the two algorithms. Figure 2 shows the characteristic plots of the convergence of lower and upper bounds for the two algorithms on instance SSLP10.50.1000 over time. Like in the SSCh case, the plots reveal how both algorithms are able to quickly converge to a near optimal solution in the first few iterations. Therefore, one can terminate the algorithm once it has found that initial stable first-stage solution. The convergence of the lower and upper bounds for the two algorithms is very similar. In this case the LPD algorithm closes the gap just a little before the D^2 algorithm.

Table 6: D^2 Computational Results for SSLP with Five Replications.

	Iters	D^2 Cuts	L^2 Cuts	CPU (secs)			CPLEX on EF	
Problem	Avg	Avg	Avg	Min	Max	Avg	CPU (secs)	Gap
SSLP5.25.50	18.33	33.33	0.67	0.08	0.22	0.17	1.73	0.00%
SSLP5.25.100	18.33	66.67	0.67	0.16	0.33	0.25	130.20	0.00%
SSLP5.50.50	16.60	0.00	0.00	0.09	0.17	0.13	1.30	0.00%
SSLP5.50.100	17.00	0.00	0.00	0.14	0.20	0.19	3.61	0.00%
SSLP10.50.50	249.67	233.33	1.00	17.17	47.44	27.75	>10800	0.30%
SSLP10.50.100	251.00	616.67	1.00	18.51	96.21	37.88	>10800	0.74%
SSLP10.50.500	268.33	2416.67	1.00	61.58	96.21	80.84	>10800	12.57
SSLP10.50.1000	268.83	5166.67	1.00	100.66	148.71	129.90	>10800	24.17
SSLP10.50.2000	274.50	9000.00	1.00	200.50	281.09	243.69	>10800	39.74
SSLP15.45.5	281.33	248.33	3.33	9.67	1555.45	306.53	>10800	0.67%
SSLP15.45.10	608.17	365.00	2.00	38.13	1400.89	581.91	>10800	0.52%
SSLP15.45.15	426.67	835.00	2.67	24.35	2006.32	552.25	>10800	1.50%
SSLP15.45.20	312.40	668.00	1.80	49.01	1613.02	454.62	>10800	1.00%
SSLP15.45.25	375.60	960.00	3.00	229.46	615.95	433.17	>10800	0.73%

D^2 algorithm stopping tolerance used is 0.0001% Gap.

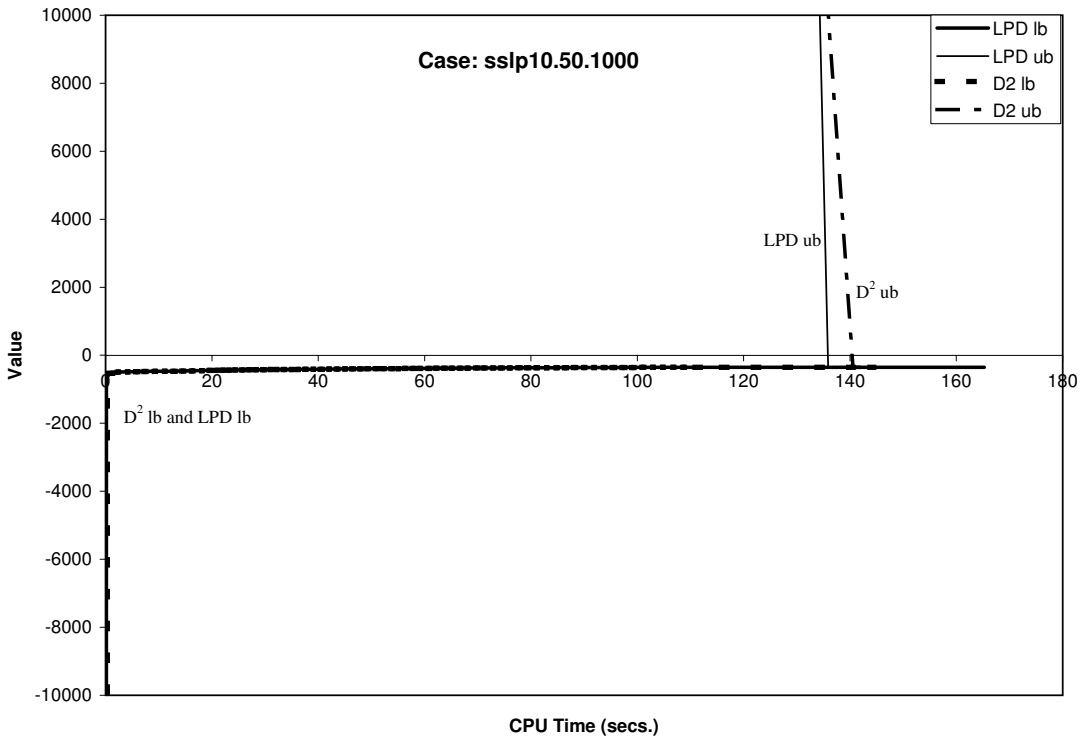

Figure 2: Convergence of lower and upper bounds for SSLP10.50.1000

The results of the *LPD* algorithm on the SSLP and the SSCH test instances are encouraging. It is important to note that the SSLP test set has characteristics that seem to be amenable to this solution approach. These particular instances have a large integrality gap (up to over 20%), but if they are solved with the first-stage binary restrictions enforced, with relaxed subproblems, that gap becomes much smaller. The effect of this is that the first-stage solution of the LP relaxation (second-stage) is often close to the optimal first-stage solution. The disjunctive cuts generated by the *LPD* algorithm are highly effective at closing the optimality gap in this situation, but might not be nearly as effective for problem instances with large optimality gaps. However, the SSCH test instances are much larger than the SSLP cases and do not have such a tight relaxation. The ability of these cutting plane algorithms to find good solutions with pretty small gaps on these instances gives evidence that disjunctive cuts are effective towards solving stochastic mixed 0-1 integer programs.

5 Conclusion

This paper presents a first computational study of a class of disjunctive cuts for two-stage stochastic mixed 0-1 programs with recourse. In particular, the paper extends and implements a cutting plane decomposition method for SMIP proposed by Carøe (1998) that uses lift-and-project cuts based on the extensive form of the two-stage stochastic

mixed 0-1 program. The advantage of this approach is that with some assumptions on where the data uncertainty appears in the problem, a cut derived for one scenario can be made valid for other scenarios, potentially reducing the solution time. A comparison with an alternative disjunctive cut algorithm for SMIP, the D^2-algorithm, is made and it is computationally shown that both cuts seem to be equally effective in solving our test instances. In general, the computational results demonstrate the effectiveness of disjunctive cuts in solving large-scale instances from the literature arising in server location under uncertainty and stochastic supply chain management. Since the cuts prove to be effective on their own, this line of research holds promise for the development of branch-and-cut algorithms for SMIP that take advantage of these disjunctive cuts. Finally, the decomposition methods presented in this paper are amenable to parallel/distributed implementation, which can significantly reduce computation time.

Acknowledgments. This research was supported by a grant (CNS-0540000) from the Dynamic Data-Driven Application Systems (DDDAS) program of the National Science Foundation. The authors would like to thank the anonymous referees for their comments which helped improve the presentation of the paper.

Appendix

SSCh Model Formulation

In this section we provide a short description and formulations of the stochastic supply chain planning problem that generates the SSCH test instances. This application of stochastic programming was reported in Alonso-Ayuso et al. (2003). The problem is to minimize the cost of running a production supply chain minus the revenue from selling products. The first-stage decisions are binary and concern plant sizing. The second-stage decisions are mixed binary and include recourse for the binary decisions from the first-stage, product allocation, vendor selection, and tactical decisions for running the created supply chain. The uncertainty in this problem arises in the objective function in the form of net profit from products and in the right-hand side in the form of demand. The two stage formulation of this stochastic MIP is given next.

Decision Variables:
x: Binary first-stage decisions concerning the expansion of facilities
$y(\omega)$: Binary second-stage decisions concerning expansion of facilities, which vendors to supply from, and which products are processes
$z(\omega)$: Continuous decision variables for the operation of the supply chain
Problem Data:
a: Cost of expanding facilities
b: Cost of expanding facilities in the second-stage
$c(\omega)$: Operations costs of the supply chain
A^1: Capacity constraint for facility expansion combined with a budget constraint

A^2: Parameters to ensure that capacity expansions are well defined and are within the budget constraint after the second-stage
B: Parameters to force the second-stage recourse decisions to be feasible with the plant capacities
C: Parameters to ensure the bill of materials requirements and that the demands are met
q: Budget constraint for capacity expansion
$p(\omega)$: Random demands for the products

$$\begin{aligned} \min \quad & a^\top x + E_{\tilde{\omega}}[f(x,\omega)] && \text{(15a)} \\ \text{s.t.} \quad & A^1 x = q && \text{(15b)} \\ & x \in \{0,1\} && \text{(15c)} \end{aligned}$$

where for any x satisfying the constraints (15b - 15c) and for each scenario realization ω of $\tilde{\omega}$ we have

$$\begin{aligned} f(x^k,\omega) = \min \quad & by(\omega) + c(\omega)z(\omega) && \text{(16a)} \\ \text{s.t.} \quad & By(\omega) + Cz(\omega) = p(\omega) - A^2 x^k && \text{(16b)} \\ & y(\omega) \in \{0,1\},\ z(\omega) \geq 0,\ \forall \omega \in \Omega && \text{(16c)} \end{aligned}$$

Notice that the first-stage decision variables are pure binary while the second-stage subproblems are mixed binary. The objective value is to minimize the cost of setting up the chain minus the revenue from selling products. The first-stage constraints include budgetary constraints and maximum expansion that can be completed as well as forcing at least some production to be done. The second-stage constraints are for the operational decisions of the supply chain e.g. which venders to buy from, which products to sell, which plants manufacture which products, etc.

SSLP Model Formulation

In this appendix we provide a short description of the application inspiring the stochastic server location problem (SSLP) test instances and a summary of the model formulation reported in Ntaimo and Sen (2004). The basis for stochastic server location problems is how to make optimal strategic decisions regarding where to locate "servers" now in the face of future uncertainty in resource demand based on whether or not a "client" will be available in the future for service. This problem arises is such diverse fields as electric power management, internet services, and telecommunications. The two-stage SSLP problem can be summarized as follows (Ntaimo and Sen, 2004):

Decision Variables:

x_j is 1 if a server is located at site j, 0 otherwise

y_{ij}^{ω} is 1 if client i is served by a server at location j under scenario ω, 0 otherwise

Problem Data:

c_j: Cost of locating a server at location j.
u: Server capacity.
v: An upper bound on the total number of servers that can be located.
w_z: Minimum number of servers to be located in zone $z \in Z$
$\mathcal{J}_z$: The subset of server locations that belong to zone z.
q_{ij}: Revenue from client i being served by server at location j.
d_{ij}: Client i resource demand from server at location j.
$h^i(\omega)$: Takes a value of 0 if client i is present in scenario ω, 0 otherwise.
p_ω: Probability of occurrence of scenario $\omega \in \Omega$.

$$\text{Min} \sum_{j \in \mathcal{J}} c_j x_j - E_{\tilde{\omega}}[f(x, \tilde{\omega})] \tag{17a}$$

$$\text{s.t.} \sum_{j \in \mathcal{J}} x_j \leq v, \tag{17b}$$

$$\sum_{j \in \mathcal{J}_z} x_j \geq w_z, \ \forall z \in \mathcal{Z}, \tag{17c}$$

$$x_j \in \{0, 1\}, \ \forall j \in \mathcal{J}, \tag{17d}$$

where for any x satisfying the constraints (17b - 17d) and for each scenario realization ω of $\tilde{\omega}$ we have

$$f(x, \omega) = \text{Min} \ -\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} q_{ij} y_{ij} + \sum_{j \in \mathcal{J}} q_{j0} y_{j0} \tag{18a}$$

$$\text{s.t.} \sum_{i \in \mathcal{I}} d_{ij} y_{ij} - y_{j0} \leq u x_j, \ \forall j \in \mathcal{J}, \tag{18b}$$

$$\sum_{j \in \mathcal{J}} y_{ij} = h^i(\omega), \ \forall i \in \mathcal{I}, \tag{18c}$$

$$y_{ij} \in \{0, 1\}, \ \forall i \in \mathcal{I}, j \in \mathcal{J}, \tag{18d}$$

$$y_{j0} \geq 0, \ \forall j \in \mathcal{J}. \tag{18e}$$

The objective function in (17a) is to maximize expected profit from providing some limited resource to the clients. Constraints (17b) enforces the requirement that only up to a total of v available servers can be installed while constraints (17c) are the zonal requirements of having at least w_z servers located in a given zone $z \in \mathcal{Z}$. Constraints (17d) are the binary restrictions on the strategic decision variables.

In the second-stage, constraints (18b) require that a server located at site j serve only up to its capacity u. The continuous variable y_{j0} accommodate any overflows that are not served due to limitations in server capacity and result in a loss of revenue at a rate of q_{j0}. Constraints (18c) meets the requirement that each available client be served by only one server. Finally, constraints (18d) are the binary restrictions on the tactical decision variables and constraints (18e) are nonnegativity restrictions on the overflow variables.

References

S. Ahmed and R. Garcia. Dynamic capacity acquisition and assignment under uncertainty. *Annals of Operational Research*, 124:267–283, 2003.

A. Alonso-Ayuso, L.F. Escudero, A. Garín, M.T. Ortuño, and G. Perez. An approach for strategic supply chain planning under uncertainty based on stochastic 0-1 programming. *Journal of Global Optimization*, 26:97–124, 2003.

E. Balas. Disjunctive programming: cutting planes from logical conditions. In O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, editors, *Nonlinear Programming 2*. Academic Press, 1975.

E. Balas. Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.

E. Balas. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algorithmic and Discrete Mathematics*, 6:466–486, 1984.

E. Balas, E. S Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 integer programs. *Mathematical Programming*, 58:295–324, 1993.

E. Balas, E.S. Ceria, and G. Cornuéjols. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42:1229–1246, 1996.

C. Blair and R. Jeroslow. A converse for disjunctive constraints. *Journal of Optimization Theory and Applications*, 25:195–206, 1978.

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

Claus C. Carøe. *Decomposition in Stochastic Integer Programming*. Ph.D. thesis, Dept. of Operations Research, University of Copenhagen, Denmark, 1998.

Inc. ILOG. *CPLEX 9.0 Reference Manual*. ILOG CPLEX Division, 2003.

G. Laporte and F. V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 1:133–142, 1993.

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

L. Ntaimo. Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. *Submitted*, 2006. http://ie.tamu.edu/people/faculty/Ntaimo/default.htm.

L. Ntaimo and S. Sen. The million-variable 'march' for stochastic combinatorial optimization. *Journal of Global Optimization*, 32(3):385–400, 2004.

L. Ntaimo and S. Sen. A comparative study of decomposition algorithms for stochastic combinatorial optimization. *Computational Optimization and Applications Journal*, 2006. To appear.

S. Sen and J.L. Higle. The C3 theorem and a D2 algorithm for large scale stochastic mixed-integer programming: Setconvexification. *Mathematical Programming*, 104(1): 1–20, 2005.

S. Sen and H.D. Sherali. Nondifferentiable reverse convex programs and facetial cuts via a disjunctive characterization. *Mathematical Programming*, 37:169–183, 1987.

S. Sen and H.D. Sherali. Decomposition with branch-and-cut approaches for two stage stochastic mixed-integer programming. *Mathematical Programming*, 106(2):203–223, 2006.

H.D. Sherali and C.M. Shetty. Optimization with disjunctive constraints. *Lecture Notes in Economics and Math. Systems,*, 181:411–430, 1980.

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

R. J-B. Wets. Stochastic programs with fixed recourse: The equivalent deterministic problem. *SIAM Review*, 16:309–339, 1974.