Disjunctive Decomposition for Two-Stage Stochastic Mixed-Binary Programs with GUB Constraints

Brian Keller
Booz Allen Hamilton, 134 National Business Parkway, Annapolis Junction, MD 20701, USA,
keller_brian@bah.com

Güzin Bayraksan
Systems and Industrial Engineering, University of Arizona, 1127 E. James E. Rogers Way, Engineering Building #20, Tucson, AZ 85721, USA, guzinb@sie.arizona.edu

The disjunctive decomposition ($D^2$) algorithm has emerged as a powerful tool to solve stochastic integer programs. In this paper, we consider two-stage stochastic integer programs with binary first-stage and mixed-binary second-stage decisions and present several computational enhancements to $D^2$. First, we explore the use of a cut generation problem restricted to a subspace of the variables, which yields significant computational savings. Then, we examine problems with generalized upper bound (GUB) constraints in the second-stage and exploit this structure to generate cuts. We establish convergence of $D^2$ variants. We present computational results on a new stochastic scheduling problem with uncertain number of jobs motivated by companies in industries such as consulting and defense contracting where these companies bid on future contracts but may or may not win the bid. The enhancements reduced computation time on average by 45% on a set of test problems.

Key words: stochastic integer programming; disjunctive decomposition; generalized upper bound constraints; stochastic scheduling

1. Introduction

Stochastic mixed integer programs (SMIPs) comprise one of the most difficult classes of optimization problems as they combine the large-scale nature of stochastic programs with the computational challenges of integer programming. Throughout the paper, we consider the following SMIP:

$$\min_{x \in X \cap \mathbb{B}} cx + E[f(x, \tilde{\omega})],$$

where $X = \{x \in \mathbb{R}^{d_x} : Ax \geq b, x \geq 0\}$, $\mathbb{B} \subset \mathbb{R}^{d_x}$ is the set of binary vectors with dimension $d_x$, $\tilde{\omega}$ is a random vector whose distribution is assumed known and does not depend on $x$. 
The support of $\tilde{\omega}$ is denoted by $\Omega$ and a realization of $\tilde{\omega}$ is denoted by $\omega$. $E$ is the expectation operator and expectation in (1) is taken with respect to the distribution of $\tilde{\omega}$. For any $\omega \in \Omega$,

$$f(x, \omega) = \min_y qy$$

s.t. \begin{align*}
Wy &\geq r(\omega) - T(\omega)x, \\
y &\geq 0, y_j \in \{0, 1\} \text{ for } j \in B,
\end{align*} \hspace{1cm} (2)

where $B$ is an index set specifying the binary variables. Problem (2) is referred to as the second-stage subproblem. For a given $(x, \omega) \in (X \cap B, \Omega)$, it is a 0-1 mixed-integer program (MIP). Throughout the paper, we refer to (2) as MIP subproblems. Above, $y$ is the decision vector corresponding to a particular scenario subproblem and hence depends on $\omega$. However, we suppress this in (2). In the rest of the paper, we use $\{y(\omega)\}_{\omega \in \Omega}$ interchangeably to represent the subproblem decisions. To simplify our analysis, we restrict our attention to subproblems with deterministic $q$ and $W$ although concepts presented here may be applied to problems with random cost vectors, $q(\omega)$, and random recourse matrix, $W(\omega)$, with appropriate technical modifications (Ntaimo, 2010). For all $(x, \omega) \in (X \cap B, \Omega)$, we denote the set of mixed-integer feasible solutions of (2) as $Y(x, \omega) = \{y : Wy \geq r(\omega) - T(\omega)x, y \geq 0, y_j \in \{0, 1\} \text{ for } j \in B\}$, and denote the linear programming (LP) relaxation of $Y(x, \omega)$ as $Y_{LP}(x, \omega)$. We assume the following:

A1. $\Omega$ is a finite set,

A2. For all $(x, \omega) \in (X \cap B \times \Omega)$, $Y(x, \omega)$ is nonempty.

Assumption A2 implies $f(x, \omega) < \infty$ for all $(x, \omega) \in (X \cap B \times \Omega)$, which is the relatively complete recourse assumption of the stochastic programming literature. This condition may be assured by adding a variable with arbitrarily high cost to each constraint in the subproblem to ensure feasibility.

The integer variables in the second stage lead to a nonconvex and discontinuous recourse function, $E[f(x, \tilde{\omega})]$, that is often quite difficult to optimize (Schultz, 1993). As a result, algorithm design for SMIP is quite challenging; however, much progress has been made in recent years; see, e.g., the surveys by Klein Haneveld and van der Vlerk (1999), Schultz (2003), and Sen (2005). One of the earliest algorithms for SMIP is the integer L-shaped method of Laporte and Louveaux (1993). This algorithm solves subproblems as MIPs and generates optimality cuts in a fashion similar to Benders decomposition (Benders, 1962). Despite the nonconvexity of the second-stage value function, Laporte and Louveaux (1993) provide linear cuts that are exact at the binary solution they are created and are lower-approximations of $E[f(x, \tilde{\omega})]$ at other binary solutions. The linearity of the cuts stems from
the fact that $x$ are binary. As the cuts are valid for a general class of second-stage problems, they can be weak. Note also that the second-stage MIP subproblems need to be solved to optimality to obtain a valid cut.

The disjunctive decomposition ($D^2$) algorithm developed by Sen and Higle (2005) uses ideas from disjunctive programming (Balas, 1975; Blair and Jeroslow, 1978; Sherali and Shetty, 1980) to convexify the feasible regions of MIP subproblems. An improving sequence of LP relaxations of subproblems are solved, and this is embedded in a Benders decomposition framework (see §2 for more details). Sen and Sherali (2006) extend $D^2$ to approximate the MIP subproblem value functions, $f(x, \tilde{\omega})$, and call this algorithm $D^2$-BAC (disjunctive decomposition with branch-and-cut). In $D^2$-BAC, in addition to MIP subproblem feasible region convexification, disjunctions are formed on the optimal objective function values at the nodes in the partially solved branch-and-bound tree and a disjunctive cut on the MIP value function is passed to the master problem. Ntaimo and Sen (2008) provide a comparison of $D^2$ and $D^2$-BAC noting that while both algorithms outperform direct solution and the integer L-shaped algorithm of Laporte and Louveaux (1993), neither $D^2$ nor $D^2$-BAC consistently solve faster than the other. We note that $D^2$ forms disjunctions in the $\{y(\omega)\}_{\omega \in \Omega}$-space. Alternative cut generations utilize the $\{x, y(\omega)\}_{\omega \in \Omega}$-space. This was first developed by Carøe (1998) and Carøe and Tind (1997) and extensions and computational results are presented in Ntaimo and Tanner (2008). Recently, variants of $D^2$ have been developed to accommodate continuous first-stage variables (Ntaimo and Sen, 2007) and random recourse matrix (Ntaimo, 2010), and computational speed-ups have been developed for the cut generation phase of the algorithm (Yuan and Sen, 2009). This paper continues in this vein of research and provides further computational enhancements to $D^2$ and explores disjunctions based on special structures that appear in many SMIPs.

Yuan and Sen (2009) note that the $D^2$ cut generation linear program (CGLP) is a stochastic linear program and realize significant improvement in computation times by applying the L-shaped method of van Slyke and Wets (1969) to solve it. Following the spirit of Yuan and Sen (2009), we investigate the use of computational speed-ups in the CGLP. For deterministic MIPs, Balas et al. (1993) generate disjunctive cuts restricted to a subspace of the variables. We adapt the restricted CGLP to the $D^2$ algorithm and find significant improvements in computation times.

Previous $D^2$ research has focused on using disjunctive cuts formed on binary disjunctions i.e., $(y_j \leq 0) \cup (y_j \geq 1)$ for $j \in B$ over the second stage feasible set. In this paper, we
investigate the use of disjunctions on generalized upper bound (GUB) constraints in the MIP subproblems. GUB constraints take the form of $\sum_{j \in G} y_j = 1$ with $y_j \in \{0, 1\}$ for $j \in G \subseteq B$. The GUB structure often contains valuable information that can be used to increase effectiveness of algorithms. For example, GUB cover cuts (Wolsey, 1990) are much more effective than regular cover cuts for problems with GUB structures. Also, in branch-and-bound for deterministic MIP problems, branching on GUB constraints can be more effective than standard 0-1 branching for many problems (Wolsey, 1998). To realize similar computational speed-ups, we use disjunctions on GUB sets to convexify the MIP subproblem feasible regions. Our initial computational results suggest that disjunctions formed on GUB sets can be more effective than disjunctions formed on 0-1 variables for MIP set convexification within the $D^2$ algorithm.

GUB constraints appear in many stochastic optimization problems. One application that motivated the work presented in this paper is a stochastic scheduling problem with an uncertain number of jobs faced by companies in industries such as consulting, engineering services, and defense contracting. These companies bid on future contracts but may or may not win the contract, resulting in an uncertain number of jobs. In this context, GUB constraints ensure that jobs with winning bids are scheduled in the second-stage when this information is revealed (see §4.1 for details). Another motivating example arises in infrastructure planning of water reuse systems. Water reuse systems carry water from wastewater treatment plants to users of treated water such as golf courses and public parks. In this setting, GUB constraints allow selection of pipe diameters and pump sizes among a discrete set of options (Zhang et al., 2010). Needless to say, GUB constraints arise in many other important applications. In this paper, we model the aforementioned stochastic scheduling problem as a two-stage SMIP and test the effectiveness of our enhancements on instances of this problem.

The remainder of the paper is organized as follows. In §2, we review the $D^2$ algorithm and in §3, we describe several computational enhancements to $D^2$. In §4, we introduce the stochastic scheduling problem with uncertain number of jobs and present computational results. We conclude in §5 with a summary and future research directions.

2. Disjunctive Decomposition

In this section, we first give an overview of disjunctive decomposition and then discuss the cut generation phase of the algorithm in more detail. In the rest of the paper, we will refer
back to these details to explain our enhancements to $D^2$.

2.1. Overview of $D^2$

The $D^2$ algorithm is a decomposition-based algorithm similar to Benders decomposition (Benders, 1962), which is referred to as the L-shaped method (van Slyke and Wets, 1969) in stochastic programming literature. The L-shaped method uses convexity of $E[f(x, \tilde{\omega})]$ over $X$ when the second-stage decisions are continuous to generate lower-approximations of $E[f(x, \tilde{\omega})]$. Then, this lower-approximation plus the first-stage objective function ($cx$) is optimized over the first-stage feasible region to generate a candidate solution and to obtain a lower bound on the optimal objective function value. This problem is referred to as the master problem. The value of $E[f(x, \tilde{\omega})]$ at the current candidate solution is obtained by solving the second-stage subproblem, which decomposes into separable problems for each $\omega \in \Omega$. This, together with the first-stage objective function evaluated at the candidate solution yields an upper bound on the optimal value. Subproblems are also used to form a subgradient of $E[f(x, \tilde{\omega})]$, which provides a refined lower-approximation by adding a "cut" to the master problem. The algorithm continues until upper and lower bounds are sufficiently close. Variations to this basic framework include feasibility cuts to eliminate $x$ that cause infeasibility in the second-stage and adding multiple cuts (Birge and Louveaux, 1997).

The $D^2$ algorithm expands on the L-shaped method, making it possible to bypass the difficulties posed by having integrality restrictions in the second-stage—such as nonconvex and discontinuous $E[f(x, \tilde{\omega})]$—while still allowing for computational advantages such as decomposition, and LP (relaxation) solutions. In the SMIP setting, the second-stage subproblems now contain integrality restrictions. The $D^2$ algorithm, instead of solving them as MIPs as in (Laporte and Louveaux, 1993), solves their LP relaxations. This provides a significant computational advantage. If the subproblem solution does not satisfy integrality, disjunctive cuts of the form $\pi^T y \geq \pi_0(x, \omega)$ are generated. By the addition of disjunctive cuts, the current fractional solution $y$ is eliminated and thus, the LP relaxations of the subproblems are sequentially refined throughout the $D^2$ algorithm.

Note that constraints $Wy \geq r(\omega) - T(\omega)x$ in subproblems (2) are affine linear in $x$. This fact and hence the convexity of $f(x, \omega)$ in $x$ is exploited in the L-shaped method to develop convex lower-approximations of $E[f(x, \tilde{\omega})]$. However, the right-hand-side of the disjunctive cuts $\pi_0(x, \omega)$ is a piecewise linear concave function in $x$. As a result, lower approximations of $f(x, \omega)$ obtained by solving this LP relaxation will be nonconvex in general. Fortunately,
it is possible to form a function, denoted $\pi_c(x, \omega)$, that is affine linear in $x$. $\pi_c(x, \omega)$ is equal to $\pi_0(x, \omega)$ when $x$ is binary and provides a lower-approximation to $\pi_0(x, \omega)$ when $0 < x < 1$. Then the cuts $\pi^T y \geq \pi_c(x, \omega)$ are used instead of $\pi^T y \geq \pi_0(x, \omega)$ to convexify the subproblems.

The disjunctive cuts $\pi^T y \geq \pi_0(x, \omega)$ are formed in a similar manner as in deterministic MIPs, by solving a cut generation linear program (CGLP) (see §2.2). The translation of $\pi_0(x, \omega)$ to $\pi_c(x, \omega)$ is done by solving a set of LPs for each $\omega \in \Omega$, called the right-hand-side linear programs (RHSLPs). RHSLPs are derived from reverse convex programming that uses a disjunctive characterization to form facets of the convex hull of reverse convex sets (Sen and Sherali, 1985). For details on the RHSLPs, see (Sen and Higle, 2005).

With the addition of the cuts $\pi^T y \geq \pi_c(x, \omega)$ where $\pi_c(x, \omega)$ is affine linear in $x$, we are back in the algorithmic setting of the L-shaped method. We can now pass an L-shaped optimality cut to the master problem and are ensured that this provides a lower-approximation to $E[f(x, \tilde{\omega})]$. A summary of $D^2$ is given in Algorithm 1 (for a definition of $\rho_i^c(x^i, \omega)$, see §2.2). Note that step 2 provides disjunctive cuts in two phases as explained above: First, by solving a CGLP to obtain $\pi^T y \geq \pi_0(x, \omega)$ (step 2.1) and then by solving RHSLPs to obtain $\pi_c(x, \omega)$ (step 2.2). Significant computational effort is spent in cut generation and therefore our enhancements focus on the CGLP phase of $D^2$. Next, we review this phase of $D^2$.

### 2.2. Cut Generation

At each iteration of $D^2$, LP relaxations of MIP subproblems are solved. If the subproblem solution does not satisfy integrality, disjunctive cuts of the form $\pi^T y \geq \pi_0(x, \omega)$ are generated to convexify the subproblem. Under the fixed recourse assumption, the left-hand-side coefficients, $\pi$, remain the same for all $\omega \in \Omega$. This result is known as the Common Cut Coefficient ($C^d$) Theorem (Sen and Higle, 2005). This allows for a cut generated by one scenario subproblem to be easily translated into a cut that is valid for another scenario subproblem. The common cut coefficients, $\pi$, are found by solving the CGLP. This is done in a similar way as the lift-and-project cut generation for deterministic MIPs, except now the CGLP becomes a two-stage stochastic linear program with simple recourse. As in deterministic MIPs, the feasible region of this stochastic linear program provides a family of cuts and the objective function aims to find a cut that is most desirable. A commonly used objective is to find the “deepest” cut, i.e., a one that cuts off the optimal vertex of the current relaxation by more than any member of the family. In the SMIP setting, there could be more than
Algorithm 1 Disjunctive Decomposition (D²)

0. **Initialize.** Let \( \epsilon > 0 \) and \( x^1 \in X \cap \mathbb{B} \) be given. Let \( i \leftarrow 1 \) and initialize an upper bound \( V_0 = \infty \) and a lower bound \( v_0 = -\infty \). Set \( W^1 \leftarrow W \), \( T^1(\omega) \leftarrow T(\omega) \), \( r^1(\omega) \leftarrow r(\omega) \), and \( \rho_c^i(x^1, \omega) \leftarrow r^1(\omega) - T^1(\omega)x^1 \).

1. **Solve LP Subproblems.** Set \( V_i \leftarrow V_{i-1} \). For each \( \omega \in \Omega \), solve subproblem LP relaxations (3) using \( W^i \) and \( \rho_c^i(x^i, \omega) \). If \( \{y^i(\omega)\}_{\omega \in \Omega} \) satisfies integrality, set \( V_i \leftarrow \min\{cx^i + E[f(x^i, \tilde{\omega})], V_i\} \) and go to step 4.

2. **Generate Cuts.**

2.1. **Solve CGLP.** Choose disjunction variable \( j(i) \in B \) and obtain \( \pi^i \) by solving CGLP formulated using original subproblem data appended with cuts generated from variables with indices less than \( j(i) \) to guarantee convergence. (See §3.3.) Define \( W^{i+1} \) by appending \( \pi^i \) to \( W^i \).

2.2. **Solve RHSLPs.** For each \( \omega \in \Omega \), solve RHSLP to obtain \( \pi^i_c(x^i, \omega) \). Define \( r^{i+1}(\omega) \) and \( T^{i+1}(\omega) \) by appending elements of \( \pi^i_c(x^i, \omega) \) to \( r^i(\omega) \) and \( T^i(\omega) \). Update \( \rho_c^{i+1}(x^i, \omega) \leftarrow r^{i+1}(\omega) - T^{i+1}(\omega)x^i \).

3. **Update and Solve LP Subproblems.** For each \( \omega \in \Omega \), solve the updated LP subproblem (3) with \( W^{i+1} \) and \( \rho_c^{i+1}(x^i, \omega) \). If \( \{y^i(\omega)\}_{\omega \in \Omega} \) satisfies integrality, set \( V_i \leftarrow \min\{cx^i + E[f(x^i, \tilde{\omega})], V_i\} \).

4. **Update and Solve the Master Problem.** Using the dual multipliers from the most recently solved subproblems (either step 1 or step 3), add L-shaped optimality cut to the master. Obtain \( x^{i+1} \) by solving the MIP master problem and let \( v_i \) be the optimal value of the master. If \( V_i - v_i \leq \epsilon \cdot \max\{|V_i|, |v_i|\} \), stop. Otherwise, \( i \leftarrow i + 1 \) and go to step 1.

one scenario subproblem with the same element of \( y \) fractional. Therefore, the objective of the CGLP can be set to maximize the “expected” depth of the cut among all such scenario subproblems.

For a given \( x^i \) at iteration \( i \), define the scenario subproblem LP relaxation as

\[
f^i_c(x^i, \omega) = \min_{\substack{qy \\ W^iy \geq \rho^i_c(x^i, \omega), \\ y \geq 0}} \{qy\}
\]

where \( \rho^i_c(x^i, \omega) = r^i(\omega) - T^i(\omega)x^i \). At the first iteration of \( D^2 \), the first set of constraints in (3) is equivalent to those of (2), i.e., \( W^1 = W \), \( r^1(\omega) = r(\omega) \), and \( T^1(\omega) = T(\omega) \). At subsequent iterations, \( W^i \) also contains the cut coefficients \( \pi \) generated during the algorithm and \( \rho^i_c(x^i, \omega) \) also includes elements of \( \pi_c(x, \omega) \) generated from the RHSLPs.

In step 1 at iteration \( i \) of \( D^2 \), if a fractional solution to (3) exists for any scenario, we go to step 2 and add disjunctive cuts to eliminate this fractional solution and convexify the
subproblems. Let $j(i) \in B$ denote an index $j$ for which $y^i_j(\omega)$ is fractional for some $\omega \in \Omega$. To eliminate this fractional solution, a disjunction of the form $S^i(x^i, \omega) = S_{0,j(i)}(x^i, \omega) \cup S_{1,j(i)}(x^i, \omega)$, where

$$S_{0,j(i)}(x^i, \omega) = \{y \geq 0 : W^i y \geq \rho^i C(x^i, \omega), \quad -y_j(i) \geq 0\},$$

$$S_{1,j(i)}(x^i, \omega) = \{y \geq 0 : W^i y \geq \rho^i C(x^i, \omega), \quad y_j(i) \geq 1\}$$

is used. Before proceeding, we note that (4a) and (4c) might not use $W^i$ and $\rho^i C(x^i, \omega)$ but rather only a subset of their elements as noted in step 2.1 in Algorithm 1 (see also §3.3). However, for ease of exposition, here, we will use $W^i$ and $\rho^i C(x^i, \omega)$. Let $\lambda_{01}$ denote the vector of multipliers associated with the right-hand-side constraints in (4a) and $\lambda_{02}$ the scalar multiplier associated with the constraint in (4b). Similarly, let $\lambda_{11}$ and $\lambda_{12}$ denote the multipliers associated with the right-hand-side constraints in (4c) and (4d), respectively. Define

$$I^i_j = \begin{cases} 1, & \text{if } j = j(i) \\ 0, & \text{otherwise,} \end{cases}$$

and let $W^i_j$ denote the $j^{th}$ column of $W^i$. Then the CGLP is the stochastic version of the LP used to generate the lift-and-project cuts. This problem can be viewed as a two-stage stochastic linear program (Yuan and Sen, 2009), and we present it below in this form.

\textbf{CGLP:} \begin{align*}
\min_{\pi, \lambda} & \quad E[y^i(\tilde{\omega})] \pi + E[-\pi_0(\lambda, \tilde{\omega})] \\
\text{s.t.} & \quad \pi_j \geq \lambda_{01} W^i_j - I^i_j \lambda_{02}, \quad \forall j, \\
& \quad \pi_j \geq \lambda_{11} W^i_j + I^i_j \lambda_{12}, \quad \forall j, \\
& \quad -1 \leq \pi_j \leq 1, \quad \forall j, \\
& \quad \lambda_{01}, \lambda_{02}, \lambda_{11}, \lambda_{12} \geq 0,
\end{align*}

where

$$-\pi_0(\lambda, \omega) = \min_z -z \begin{cases} \text{s.t.} & \quad -z \geq -\lambda_{01} \rho^i C(x^i, \omega), \\
& \quad -z \geq -\lambda_{11} \rho^i C(x^i, \omega) - \lambda_{12}, \\
& \quad -1 \leq z \leq 1. \end{cases}$$
The CGLP given in (5) aims to maximize the expected depth of the cut generated, where depth of the cut for scenario \( \omega \in \Omega \) is given by violation \( \pi^T y^i(\omega) - \pi_0(x^i, \omega) \geq 0 \). The objective in (5a) is typically a conditional expectation, taken over the set of scenarios \( \omega \in \Omega \) such that \( y^i(\omega) \) is fractional. First-stage decisions are \( \pi \) and \( \lambda \) and recourse decisions determine \( \pi_0(\lambda, \omega) \). Note that \( \pi_0(\lambda, \omega) \) in (5) is the \( \pi_0(x, \omega) \) of the disjunctive cut \( \pi^T y \geq \pi_0(x, \omega) \). In (5), the alternative notation \( \pi_0(\lambda, \omega) \) is used to present it as a two-stage stochastic linear program with first-stage decisions \( \lambda \).

The CGLP can be solved by the L-shaped method. The simple structure of the second-stage allows forming feasibility and optimality cuts without requiring an LP solver; for details see (Keller, 2009) and (Yuan and Sen, 2009). Solving the CGLP via the L-shaped method yields significant reductions in total computation time; see, e.g., the computations by Yuan and Sen (2009). In this paper, our “base” \( D^2 \) implementation (denoted \( D^201 \) in §4) solves the CGLP via the L-shaped method and we further reduce computation time on average by 45% on a set of test problems, establishing new benchmarks.

3. \( D^2 \) with Restricted CGLP and GUB Disjunctions

In this section, we discuss several enhancements to \( D^2 \). First, in §3.1, we explore generation of cuts using a restricted CGLP. We then introduce disjunctions formed on GUB sets and describe how to incorporate GUB disjunctions into the \( D^2 \) algorithm in §3.2. Finally, in §3.3, we discuss cut management and establish convergence of \( D^2 \) with GUB disjunctions and with the restricted CGLP.

3.1. Restricted Cut Generation Problem

The CGLP becomes quite large for even two disjunctions. For instance at iteration \( i \) of \( D^2 \), the CGLP has more than twice the nonzeros of \( W^i \). As disjunctive cuts are added in \( D^2 \) iterations, and hence, as \( W^i \) grows larger, the CGLP can become a bottleneck. However, it is possible to generate disjunctive cuts from a smaller LP by working in the subspace defined by the fractional components of \( y \) and lifting the cut into the original space. This method was first used in a cutting plane algorithm (Balas et al., 1993) and later in branch-and-cut (Balas et al., 1996). Here, we extend it to \( D^2 \). Our discussion in this section will be focused on the 0-1 disjunctions presented in Section 2. Later, as we introduce alternative disjunctions based on GUB constraints, we will discuss necessary modifications.
Recall that $B$ is an index set identifying the binary variables in the second-stage and let $C$ be the index set identifying the remaining continuous variables. The set $B \cup C$ contains the indices of all decision variables in (2). We will work with a subset of these variables and denote the index set for this subset as $\mathcal{R} \subseteq (B \cup C)$. Let $\{y^i(\omega)\}_{\omega \in \Omega}$ be the optimal solution to the LP relaxation of (2) for a given $x^i$ at iteration $i$ of $D^2$. Set $\mathcal{R}$ is formed as follows.

$$\mathcal{R} = \{j \in B : 0 < y^i_j(\omega) < 1 \text{ for at least one } \omega \in \Omega\} \cup \{j \in C : y^i_j(\omega) > 0 \text{ for at least one } \omega \in \Omega\} \cup \{j \in B : y^i_j(\omega) \text{ is in a disjunction}\}.$$  

(6)

In words, the set $\mathcal{R}$ contains indices of all binary variables that are fractional in at least one scenario and only indices of continuous variables that are positive in at least one scenario. Finally, $\mathcal{R}$ must include all indices used to form the disjunction to guarantee a valid cut. When 0-1 disjunctions are used as in (5), the index $j(i) \in B$ corresponding to the fractional variable $y_{j(i)}$ is the only index used to form a disjunction at iteration $i$, which is already included in $\mathcal{R}$. However, below we will use disjunctions on the GUB structure, and these disjunctions may also contain binary variables that are 0 in the current solution. Nevertheless, indices of these variables must be included in $\mathcal{R}$ to ensure the validity of the generated cut from the restricted CGLP.

We will work with a vector $y^R$ that only contains second-stage decision variables indexed by $\mathcal{R}$. If a variable is not contained in $y^R$, then we can assume it is zero since for those $j \in B$ such that $y^i_j(\omega) = 1$ for all $\omega \in \Omega$, we can complement the variable by setting column $W^j$ to $-W^j$ and setting $\rho^i(x^i, \omega)$ to $\rho^i(x^i, \omega) - W^j$, $\omega \in \Omega$. Let $\pi^R y^R \geq \pi_0(x, \omega)$ be the cut generated in the subspace. Cut generation can be done in a similar way as discussed in Section 2.2 by restricting the CGLP to use only columns indexed by $\mathcal{R}$. Next, we need to lift this cut to the space of the original second-stage variables. The right-hand-side $\pi_0(x, \omega)$ remains the same but we need cut coefficients for variables that were ignored during the restricted CGLP. This can be done as follows:

$$\pi_j = \begin{cases} 
\pi^R_j & \text{if } j \in \mathcal{R} \\
\max\{\lambda_{01}W^j, \lambda_{11}W^j\} & \text{if } j \notin \mathcal{R},
\end{cases}$$  

(7)

which ensures that $\pi_j$ satisfies (5b) and (5c).

When $j \notin \mathcal{R}$, the lifted cut coefficients $\pi_j$ may be outside the normalization range of $[-1, 1]$ specified in (5d). However, Balas et al. (1993) show finite termination of the cutting plane algorithm when (5d) is replaced with $-1 \leq \pi_j \leq 1$ for $j \in \mathcal{R}$. Working with the
restricted CGLP eliminates many rows and columns from the CGLP resulting in significant performance improvements, which we report in §4.

The cut \( \pi y \geq \pi_0(x, \omega) \) with coefficients obtained from (7) can be strengthened by using the integrality conditions on variables other than those used to form the disjunction. The following result, adapted to the \( D^2 \) framework, was originally shown by Balas (1979) and Balas and Jeroslow (1980) in the context of deterministic MIPs.

**Proposition 1.** Let \( \pi_0(x, \omega), \lambda_{01}, \lambda_{02}, \lambda_{11}, \) and \( \lambda_{12} \) be found by solving the restricted CGLP in (5) using only columns indexed by \( R \), and let \( \pi \) be obtained through (7). Then, the inequality \( \gamma y \geq \pi_0(x, \omega) \) is valid for subproblem with \((x, \omega) \in (X \cap \mathcal{B}, \Omega)\), where

\[
\gamma_j = \min \{ \lambda_{01} W_j^i + \lambda_{02} \bar{m}_j, \lambda_{11} W_j^i - \lambda_{12} \lfloor \bar{m}_j \rfloor \}, \quad j \in B,
\]

\[
\gamma_j = \max \{ \lambda_{01} W_j^i, \lambda_{11} W_j^i \}, \quad j \in C,
\]

and

\[
\bar{m}_j = \frac{\lambda_{11} W_j^i - \lambda_{01} W_j^i}{\lambda_{02} + \lambda_{12}}.
\]

**Proof.** Follows from Theorem 2.2 in Balas et al. (1996). \( \square \)

The strengthening procedure is performed after the CGLP has been solved, i.e., \( \lambda_{01}, \lambda_{02}, \lambda_{11}, \) and \( \lambda_{12} \) have already been determined. Since the strengthened cuts take the form \( \gamma y \geq \pi_0(x, \omega) \), the smallest possible coefficients \( \gamma_j, j \in B \cup C \) will lead to tighter cuts. The parameter \( \bar{m}_j \) is chosen to make \( \gamma_j \) as small as possible for \( j \in B \). Note that \( \gamma \leq \pi \), and the strengthening procedure attempts to find \( \gamma \neq \pi \). When \( \lambda_{02} = \lambda_{12} = 0 \), the strengthening procedure cannot be applied.

### 3.2. GUB Disjunctions

We now turn our attention to forming disjunctions based on GUB constraints in the second stage. Consider a GUB constraint of the form \( \sum_{j \in G} y_j = 1 \) with \( y_j \in \{0, 1\} \) for \( j \in G \subseteq B \). Let \( j_1, j_2, \ldots, j_{|G|} \) be an ordering of the variables in \( G \) and specify \( G_0 = \{ j_i : i = 1, \ldots, \nu \} \) and \( G_1 = \{ j_i : i = \nu + 1, \ldots, |G| \} \), where \( \nu \) can be chosen according to

\[
\nu = \arg\min_{t \in G} \left| \sum_{i=1}^{t} y_{j_i} - 0.5 \right|.
\]

Later, we present another way to select \( \nu \) in the context of our stochastic scheduling problem (see §4.3).
A GUB disjunction on the GUB set is

$$\left( \sum_{j \in G_0} y_j \leq 0 \right) \cup \left( \sum_{j \in G_1} y_j \leq 0 \right).$$

(9)

Following the disjunctive cut principle, to form GUB cuts, (4b) is replaced with $$- \sum_{j \in G_0} y_j \geq 0$$ and (4d) is replaced with $$- \sum_{j \in G_1} y_j \geq 0$$. Appropriately dimensioned multipliers are again denoted by $$\lambda_{01}, \lambda_{02}, \lambda_{11},$$ and $$\lambda_{12}$$. At iteration $$i$$ of $$D^2$$, let

$$I^i_{uj} = \begin{cases} 1 & \text{if } j \in G_u \\ 0 & \text{otherwise} \end{cases}$$

for $$u = 0, 1$$. Then, the GUB cuts can be generated using the same CGLP given in (5) with the following modifications:

1. Replace constraints (5b) and (5c) with

$$\pi_j \geq \lambda_{01} W^i_j - I^i_{0j} \lambda_{02}, \quad \forall j,$$

and

$$\pi_j \geq \lambda_{11} W^i_j - I^i_{1j} \lambda_{12}, \quad \forall j,$$

respectively.

2. Remove $$\lambda_{12}$$ from the CGLP subproblem constraints (5h).

This results in a modest change in the L-shaped method to solve the CGLP; see (Keller, 2009) for details.

In many problems, including our stochastic scheduling problem with uncertain number of jobs, there can be more than one GUB constraint. It is possible to form disjunctions based on more than one GUB constraint. This results in a stronger cut; however, the CGLP grows as more disjunctions are used to form the cut. We have implemented a variant of $$D^2$$ using disjunctions based on two GUB constraints. Our computational experiments on the stochastic scheduling problem indicate that the time required to generate these alternative cuts outweigh any advantage of generating stronger cuts. Disjunctions based on more than one GUB constraint were not found to be computationally competitive and are therefore omitted for brevity.

### 3.3. Cut Management and Convergence

Without proper management of cuts it is possible that sequential cutting plane methods for mixed-integer programs such as $$D^2$$ may not converge. Convergence of cutting plane algorithms for facial disjunctive programs (FDPs) have been analyzed by Jeroslow (1980)
and Blair (1980) and convergence for more general classes of problems by Sen and Sherali (1985). The implication of these results and how to achieve convergence for a lift-and-project cutting plane algorithm can be found in (Balas et al., 1993) and for $D^2$ with 0-1 disjunctive cuts in (Sen and Higle, 2005). Our discussion here is similar and offers extensions for the restricted CGLP and GUB disjunctions. We start by reviewing FDPs and convergence of $D^2$ with 0-1 disjunctions.

Recall that $Y(x, \omega)$ denotes the set of mixed-integer feasible solutions of (2) for a given $(x, \omega) \in (X \cap \mathbb{B}, \Omega)$ and $Y_{LP}(x, \omega)$ denotes its LP relaxation. We can write $Y(x, \omega)$ in conjunctive normal form as

$$Y(x, \omega) = Y_{LP}(x, \omega) \bigcap \left( \bigcup_{h \in H_s} \{ y : d_h y \geq d_{h0} \} \right),$$

(10)

where $d_h y \geq d_{h0}$ is an inequality defining a disjunction in $H_s$, $H_s$ is the set of disjunctions on logical condition $s$, and $S$ is the set of logical conditions. $Y(x, \omega)$ is facial if every inequality $d_h y \geq d_{h0}$ that appears in a disjunction of (10) defines a face of $Y_{LP}(x, \omega)$ for all $h \in H_s$, $s \in S$. For example, 0-1 MIPs are FDPs since either disjunction ($y_j \leq 0$) or ($y_j \geq 0$) for all $j \in B$ defines a face of the 0-1 MIP LP relaxation. Also, $Y(x, \omega)$ is facial if and only if for every $h \in H_s$, $s \in S$, $Y_{LP}(x, \omega) \subseteq \{ y : d_h y \leq d_{h0} \}$ (Balas, 1979). GUB disjunctions satisfy this condition since no points in $Y_{LP}(x, \omega)$ are cut off by $\sum_{j \in G_0} y_j \geq 0$ or $\sum_{j \in G_1} y_j \geq 0$ for any $G_0$ or $G_1$. Therefore, for 0-1 SMIPs with all second-stage binary variables in disjoint GUB constraints, the second-stage subproblem is a FDP.

The closure of the convex hull of FDPs can be generated via sequential convexification (Balas, 1979; Balas et al., 1993). That is, when $Y(x, \omega)$ is facial, the recursion

$$Y_s(x, \omega) = \text{conv} \left\{ \bigcup_{h \in H_s} \left( Y_{s-1}(x, \omega) \bigcap \{ y : d_h y \geq d_{h0} \} \right) \right\}, \quad s = 1, \ldots, |S|,$$

(11)

where $Y_0(x, \omega) := Y_{LP}(x, \omega)$ generates $\text{conv}\{Y_{|S|}(x, \omega)\}$ in $|S|$ steps, by sequentially generating the convex hull of each disjunction one at a time.

The $D^2$ algorithm with 0-1 disjunctions outlined in Section 2 is guaranteed to generate the convex hull of the subproblems, if necessary, in a finite number of iterations under mild conditions (Sen and Higle, 2005). This is possible because: (1) at each iteration the generated cut will cut off the current fractional solution for some $\omega \in \Omega$, and (2) there are finitely many cuts to be generated. Recall that $j(i) \in B$ denotes the index on which the 0-1 disjunction ($y_{j(i)}(\omega) \leq 0$) $\cup$ ($y_{j(i)}(\omega) \geq 1$) is formed at iteration $i$ of $D^2$. (1) is guaranteed by
using a conditional expectation in (5a) over the set of $\omega \in \Omega$ such that $y_{j(i)}(\omega)$ is fractional and by arbitrarily dropping one scenario from the conditional expectation and reoptimizing the CGLP until a nonnegative objective is found to ensure the fractional solution is cut off (Sen and Higle, 2005). (2) is guaranteed by proper management of rows appended to the $W$ matrix when forming the CGLP (see Step 2.1 of Algorithm 1). To ensure convergence, the CGLP is formed by appending to $W$ disjunctive cuts generated from variable indices $j < j(i)$ at iteration $i$. (When $j(i) = 1$, no cuts are appended.) However, unlike (11) that imposes the disjunctions one by one after fully generating all valid inequalities associated with the current disjunction, this scheme is computationally more efficient and has the same implication — that a finite number of disjunctive cuts will be required to generate $\text{conv}\{Y_{|S|}(x, \omega)\}$. The convex hull of $Y_1(x, \omega)$ can be generated in finitely many iterations because no cuts are appended when forming the CGLP due to the $j < j(i)$ index rule, i.e., with $|\Omega| < \infty$, the CGLP for $j(i) = 1$ has a finite number of extreme points corresponding to facet defining cuts for $Y_1(x, \omega)$. Once $\text{conv}\{Y_1(x, \omega)\}$ has been generated, no new cuts will be appended to $W$ when forming the CGLP for $j(i) = 2$, and thus, $\text{conv}\{Y_2(x, \omega)\}$ can be generated in finitely many iterations. Recursively, $\text{conv}\{Y_{|S|}(x, \omega)\} = Y(x, \omega)$ can be generated in finitely many iterations. Since $|X \cap B| < \infty$ and $|\Omega| < \infty$, total number of cuts required to convexify $Y(x, \omega)$ for all $(x, \omega) \in (X \cap B, \Omega)$ is finite.

Convergence of the GUB convexification similarly relies on tracking the index of the generated cuts and abiding by the $j < j(i)$ index rule when forming the CGLP. For 0-1 disjunctions, this is easy since the cut index can be the index of the disjunction variable $y_{j(i)}(\omega)$ on which the cut was formed. The GUB disjunctions use multiple variables so the same bookkeeping cannot be used. Let $\Gamma$ denote the set of disjoint GUB sets with $B = \bigcup_{G \in \Gamma} G$. We first fix an ordering $j_1, j_2, \ldots, j_{|G|}$ of the variables in each GUB set $G \in \Gamma$ and use the same ordering throughout the algorithm. Of course, we can allow for different orderings as the algorithm progresses. However, there are $\sum_{l=1}^{\lfloor \frac{|G|}{2} \rfloor} \left( \begin{array}{c} |G| \\ l \end{array} \right)$ unique ways to partition any single GUB set into two disjoint sets, and the bookkeeping can quickly get out of hand. Therefore, we use a fixed ordering. Once the ordering is established for each GUB set, we may proceed with the cut generation bookkeeping. Let $G \in \Gamma$ be the set on which we wish to form a GUB disjunction and choose $\nu$, for instance as in (8), to determine a partition $G_0, G_1$. Create a one-to-one mapping $m_1(G, \nu)$ that assigns a unique index to a combination $(G, \nu)$. We modify step 2.1 of $D^2$ (Algorithm 1) as follows:
2.1’. Solve CGLP. Choose \((G^i, \nu^i)\) to form GUB disjunction and obtain \(\pi^i\) by solving CGLP formulated using original subproblem data appended with cuts generated from indices less than \(m_1(G^i, \nu^i)\). Define \(W^{i+1}\) by appending \(\pi^i\) to \(W^i\).

With these modifications, variants of \(D^2\) can be shown to converge in a finite number of iterations.

**Proposition 2.** Consider (1) and suppose \(X = \{x \in \mathbb{R}^d : Ax \geq b, x \geq 0\}\) includes the constraints \(x \leq 1\). Assume \(A1-A2\) and \(B = \bigcup_{G \in \Gamma} G\). Define set \(\mathcal{R}\) according to (6), and follow the index rule for appending cuts to formulate the CGLP as described above for 0-1 and GUB disjunctions. Solve the CGLP and if a nonnegative objective is found, arbitrarily drop one scenario from the conditional expectation and reoptimize, repeating until a negative objective is found. Suppose the CGLP finds an extreme point solution. Then the following \(D^2\) variants will converge:

1. \(D^2\) with 0-1 cuts (\(D^2\)01),
2. \(D^2\) with 0-1 cuts and restricted CGLP (\(D^2\)01-R),
3. \(D^2\) with GUB cuts (\(D^2\)GUB),
4. \(D^2\) with GUB cuts and restricted CGLP (\(D^2\)GUB-R).

**Proof.** For proof of 1, see (Sen and Higle, 2005). We provide the proof of 2 in two steps: (i) by showing the current fractional solution will be cut off when we use the restricted CGLP, and (ii) that there are finitely many cuts to be generated. This combined with the fact that the first-stage master problem approximations converge in finitely many iterations leads to finite convergence of the \(D^2\) variant. (i) The CGLP\(^\mathcal{R}\) is formed from \(W^\mathcal{R}\), where columns \(j \notin \mathcal{R}\) are removed from \(W\). Every column removed from \(W\) results in one fewer variable and two fewer structural constraints in the CGLP\(^\mathcal{R}\), but the removed variables only appear in the constraints that have been removed. Thus, the feasible region of CGLP\(^\mathcal{R}\) is a relaxation of the feasible region of CGLP. Further, the columns removed from \(W\) correspond to variables in the CGLP objective with coefficients equal to zero. Therefore, the objective of CGLP\(^\mathcal{R}\) will be less than or equal to the objective of CGLP, and the current fractional solution will be cut off. The lifting procedure (7) does not affect the optimal objective to CGLP\(^\mathcal{R}\) since the lifted variables have objective coefficients equal to zero. (ii) See Balas et al. (1993) for finiteness of cuts to be generated when using the restricted CGLP. Since \((X \cap \mathbb{B}, \Omega)\) is finite, the total number of cuts to be generated is finite.
For proof of 3, we proceed the same way and note that (i) with \( B = \cup_{G \in \Gamma} G \), any fractional solution can be cut off by cuts based on GUB disjunctions. (ii) Given a fixed ordering of each GUB set, let \( M \) be the total number of \( (G, \nu) \) combinations of GUB cuts that can be formed. Let \( m_1(s), s = 1, 2, \ldots, M \) denote an ascending order of index \( m_1 \) with \( (G(s), \nu(s)) \) as the corresponding pair. Whenever the algorithm identifies \( (G(1), \nu(1)) \) to form a disjunction, by step 2.1’ and A2, the CGLP has a finite number of extreme points, which correspond to cut coefficients generated for any \( x \in X \cap B \). The operational assumptions ensure that a new extreme point is identified at each iteration. Once all cuts are generated, the CGLP for \( m_1(2) \) will have a fixed number of extreme points for any \( x \in X \cap B \). Continuing recursively, and noting that \( |X \cap B| < \infty, M < \infty \) and the second-stage with GUB disjunctions is facial, the sequential convexification (11) holds; therefore, by following the index rule on \( m_1 \) one can generate all cutting planes required to convexify \( Y(x, \omega) \) for all \( (x, \omega) \in (X \cap B, \Omega) \) in finitely many iterations. Proof of 4 follows from similar arguments as above.

There are three operational assumptions regarding the CGLP in Proposition 2 to ensure a finitely convergent algorithm. First is the index rule discussed above, second is the change in the conditional expectation objective to ensure the current fractional solution is cut off and the third is the assumption that the CGLP finds an extreme point solution. Extreme point solutions to the CGLP correspond to facets of the closure of \( Y_{j(i)}(x, \omega) \) at iteration \( i \) (Balas, 1979). The above proposition uses the fact that there are finitely many extreme points from finitely many polyhedra to generate the facet defining cuts. In practice, when \( x^{i+1} = x^i \), all subproblems are solved to integer optimality to obtain an upper bound and a Laporte and Louveaux (1993) type optimality cut is added to the master problem. Therefore, even if theoretical convergence cannot be shown, from a practical point of view, the addition of Laporte-Louveaux type cuts when \( x \) stabilizes ensures that the \( D^2 \) algorithm will converge since the Laporte-Louveaux algorithm converges.

4. Computational Experiments

In this section, we compare the effectiveness of the various cuts and the restricted CGLP on a new scheduling problem where the number of jobs is uncertain. We first describe the problem and instance generation, and then test the effectiveness of various enhancements.
4.1. Scheduling with Uncertain Number of Jobs

Consider a scheduling problem with an uncertain number of jobs faced by companies in industries such as consulting, engineering services, and defense contracting. These companies bid on future contracts but may or may not win the contract, resulting in a subset of jobs that may or may not require scheduling. They also have a set of jobs that are known with certainty to be scheduled. Scheduling problems with uncertain number of jobs arise in other contexts. For instance, in operating room scheduling there are typically a set of operations that must be performed. However, new operations need to be scheduled during the day as emergency surgeries arrive. Each job (or, operation) consumes multiple resources. These resources represent people with different skill sets, or equipment or capital constraints. The model formulated below allows resources to be temporarily expanded for a penalty, corresponding to costs due to outsourcing, overtime, or equipment rental. Each job has $T$ time periods in which it could be processed. In the first-stage, the known jobs are scheduled, and each known job must finish by time period $T$. At time period $T_0 \in [1,T]$, we learn which job bids have been accepted and schedule the accepted jobs so that they finish by time period $T + T_0$. During time periods $t \in [T_0 + 1, \ldots, T]$ both known jobs and jobs with accepted bids can be processed. In our computational experiments, we set $T_0 = \lceil 0.25T \rceil$. We model this problem as a two-stage SMIP using a time-indexed formulation where time is partitioned into discrete units of time $t$. Each time period $t$ starts at $t - 1$ and ends at $t$. Sets, indices, parameters and decision variables used in the model are listed in Table 1. The formulation is presented below.

\[
\min_{x,z} \sum_{j \in J} \sum_{t=1}^{T-p_j+1} c_{jt}x_{jt} + \sum_{k=1}^{K} \sum_{t=1}^{T_0} b_kz_{tk} + \sum_{\omega \in \Omega} \Pr(\omega)f(x,\omega) 
\]

\[\text{s.t.} \quad \sum_{t=1}^{T-p_j+1} x_{jt} = 1, \quad j \in J, \quad (13)\]

\[\sum_{j \in J} \sum_{s \in S(j,t)} r_{js}x_{js} - z_{tk} \leq R_k, \quad t = 1, \ldots, T_0, \quad (14)\]

\[x_{jt} \in \{0,1\}, \quad j \in J, \quad t = 1, \ldots, T-p_j+1, \quad (15)\]

\[z_{tk} \geq 0, \quad k = 1, \ldots, K, \quad t = 1, \ldots, T_0, \quad (16)\]
where

\[
 f(x, \omega) = \min_{y, u} \sum_{j \in J_B} \sum_{t = T_0 + 1}^{T + T_0 - p_j + 1} c_{jt} y_{jt} + \sum_{k = 1}^{K} \sum_{t = T_0 + 1}^{T + T_0 - p_j + 1} b_k u_{tk} \tag{17}
\]

s.t.

\[
 \sum_{t = T_0 + 1}^{T + T_0 - p_j + 1} y_{jt} = a_j(\omega), \quad j \in J_B, \tag{18}
\]

\[
 \sum_{j \in J_B} \sum_{s \in S_B(j,t)} r_{jk} y_{js} - u_{tk} \leq R_k - \sum_{j \in J} \sum_{s \in S(j,t)} r_{jk} x_{js}, \quad t = T_0 + 1, \ldots, T + T_0, \quad k = 1, \ldots, K, \tag{19}
\]

\[
 y_{jt} \in \{0, 1\}, \quad j \in J_B, \quad t = T_0 + 1, \ldots, T + T_0 - p_j + 1, \tag{20}
\]

\[
 u_{tk} \geq 0, \quad t = T_0 + 1, \ldots, T + T_0, \quad k = 1, \ldots, K. \tag{21}
\]

The objective (12) is to minimize the cost of scheduling jobs that are known with certainty plus costs due to temporary resource expansion plus the expected cost of the second-stage. The second-stage objective (17) is to minimize the cost of scheduling jobs whose bids have been accepted plus any additional temporary resource expansion. Constraints (13) require that each job is started so that it will finish by the end of time period \(T\). Constraints (14) measure the amount of temporary resource capacity required before bid acceptance is known. The second-stage constraints (18) ensure that jobs with winning bids in scenario \(\omega\) are scheduled so that they complete by the end of the rolling horizon \(T + T_0\). If the bid is rejected in scenario \(\omega\), the parameter \(a_j(\omega)\) is zero, making the constraint inactive. Constraints (19) measure the amount of temporary resource capacity required for each scenario after the bid acceptance is known. Constraints (15) and (20) force the job start decisions to be binary and constraints (16) and (21) ensure the temporary resource expansions are nonnegative.

**Remark 3.** The \(D^2\) algorithm we discussed in Section 2 is restricted to problems with only binary first-stage variables. The model presented above has continuous first-stage variables \(z_{tk}, t = 1, \ldots, T_0, k = 1, \ldots, K\). However, this does not affect \(D^2\) since the continuous variables do not affect the second-stage.
4.2. Test Problem Generation

The test problems are labeled as $A.B.C.D$ where $A \in \{5,10\}$ is the number of jobs that must be scheduled, $B \in \{10,20\}$ is the number of jobs that are bid on, $C \in \{30,50\}$ is the number of time periods $T$, and $D \in \{10,100,1000\}$ is the number of scenarios generated. All jobs have $K = 3$ resource classes. We set $T_0 = \lceil 0.25T \rceil$ and draw processing times from a discrete uniform$(1,T)$ distribution. For each bid, the probability of winning a job bid is 0.75 and is independent from other bids. Resource demands, $r_{jk}$, are generated from a discrete uniform$(1,5)$ distribution. Temporary expansion costs, $b_k$, are generated from a discrete uniform$(1,10)$ distribution. Resource capacities, $R_k$, are generated according to

$$R_k = \frac{\bar{p}^C r^C |J| + 0.75 \bar{p}^B r^B |J_B|}{(T + T_0) \rho},$$

where $\bar{p}^C = \frac{1}{|J|} \sum_{j \in J} p_j$ is a measure of average processing times for the known jobs, $\bar{r}^C_k = \frac{1}{|J|} \sum_{j \in J} r_{jk}$ is a measure of the average amount of resource $k$ consumed by the known jobs, $\bar{p}^B = \frac{1}{|J_B|} \sum_{j \in J_B} p_j$ is a measure of average processing times for jobs with bids, $\bar{r}^B_k = \frac{1}{|J_B|} \sum_{j \in J_B} r_{jk}$ is a measure of the average amount of resource $k$ consumed by the jobs with bids, and $\rho$ is a factor controlling the ratio of resources consumed to resources available. The numerator is a rough estimate of the total amount of resource $k$ we would expect to be consumed during the planning horizon, and $R_k \cdot (T + T_0)$ is the total resource available during the planning horizon. For $\rho < 1$ there tends to be enough resources for jobs to process in the horizon without exceeding $R_k$, and for $\rho > 1$ there tends to be more resources consumed than available, resulting in temporary resource expansion. We generated $\rho$ from a uniform$(0.7,1.3)$ distribution. The cost coefficients, $c_{jt}$, are set to completion time ($c_{jt} = t + p_j - 1$).

We generated a total of 12 $A.B.C.D$ combinations, shown in Table 2, and five instances for each combination. Table 2 reports the dimensions of the deterministic equivalent problem (DEP) along with the dimensions of the first-stage problem and the subproblems for each of the 12 combinations. The columns “Constr”, “Bin”, and “Cvar” give the number of constraints, binary variables, and continuous variables, respectively.

4.3. Implementation Details

The $D^2$ algorithm with each cut generation scheme was applied to the stochastic scheduling problem presented in §4.1. The algorithm was initialized by solving the LP relaxation of the stochastic MIP. All components of the fractional solution equal to zero or one were fixed at
Table 1: Stochastic scheduling problem nomenclature

Indices and Sets

- $j \in J$ set of known jobs that must be scheduled
- $j \in J_B$ set of jobs that are bid on, which may or may not require scheduling
- $t$ time periods $t = 1, 2, \ldots, T + T_0$
- $k$ resource classes $k = 1, 2, \ldots, K$
- $\omega \in \Omega$ random future scenario, $\Omega$ is the set of all future scenarios

Parameters

- $T$ number of time periods each job must be processed in
- $T_0$ time period at which we learn which job bids $j \in J_B$ are accepted
- $c_{jt}$ cost of starting job $j$ in period $t$
- $p_j$ processing time of job $j$
- $r_{jk}$ amount of resource from class $k$ consumed by job $j$ during a period
- $R_k$ total amount of resource $k$ available during a period
- $b_k$ per unit penalty for exceeding resource capacity $R_k$ in a period
- $a_j(\omega)$ 1 if bid on job $j \in J_B$ is accepted in scenario $\omega$, 0 otherwise
- $\Pr(\omega)$ probability of scenario $\omega$

Additional Sets

- $S(j,t)$ the interval of time that job $j \in J$ would be processed in if it finished in period $t$, $S(j,t) = \{\max\{1, t - p_j + 1\}, \ldots, \min\{t, T - p_j + 1\}\}$
- $S_B(j,t)$ the interval of time that job $j \in J_B$ would be processed in if it finished in period $t$, $S_B(j,t) = \{\max\{T_0 + 1, t - p_j + 1\}, \ldots, \min\{t, T + T_0 - p_j + 1\}\}$

Decision Variables

- $x_{jt}$ 1 if job $j \in J$ starts in period $t \in [1,T]$, 0 otherwise
- $z_{tk}$ amount of temporary resource expansion of resource $k$ in period $t \in [1,T_0]$
- $y_{jt}$ 1 if job $j \in J_B$ starts in period $t \in [T_0 + 1,T + T_0]$, 0 otherwise
- $u_{tk}$ amount of temporary resource expansion of resource $k$ in period $t \in [T_0 + 1,T + T_0]$

Table 2: Problem dimensions

<table>
<thead>
<tr>
<th>Problem</th>
<th>DEP</th>
<th>1st-stage</th>
<th>2nd-stage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Constr Bin Cvar</td>
<td>Constr Bin Cvar</td>
<td>Constr Bin Cvar</td>
</tr>
<tr>
<td>5.10.30.10</td>
<td>1029 1925 924</td>
<td>29 155 24</td>
<td>100 177 90</td>
</tr>
<tr>
<td>5.10.30.1000</td>
<td>100,029 177,155 90,024</td>
<td>29 155 24</td>
<td>100 177 90</td>
</tr>
<tr>
<td>5.10.50.10</td>
<td>1644 2985 1539</td>
<td>44 255 39</td>
<td>160 273 150</td>
</tr>
<tr>
<td>5.10.50.1000</td>
<td>160,044 273,255 150,039</td>
<td>44 255 39</td>
<td>160 273 150</td>
</tr>
<tr>
<td>10.10.30.10</td>
<td>1034 1952 924</td>
<td>34 132 24</td>
<td>100 182 90</td>
</tr>
<tr>
<td>10.10.30.1000</td>
<td>100,034 182,132 90,024</td>
<td>34 132 24</td>
<td>100 182 90</td>
</tr>
<tr>
<td>10.10.50.10</td>
<td>1649 2770 1539</td>
<td>49 210 39</td>
<td>160 256 150</td>
</tr>
<tr>
<td>10.10.50.1000</td>
<td>160,049 256,210 150,039</td>
<td>49 210 39</td>
<td>160 256 150</td>
</tr>
<tr>
<td>10.20.30.10</td>
<td>1134 2868 924</td>
<td>34 188 24</td>
<td>110 268 90</td>
</tr>
<tr>
<td>10.20.30.1000</td>
<td>11,034 26,988 9024</td>
<td>34 188 24</td>
<td>110 268 90</td>
</tr>
<tr>
<td>10.20.50.10</td>
<td>1749 5124 1539</td>
<td>49 204 39</td>
<td>170 492 150</td>
</tr>
<tr>
<td>10.20.50.1000</td>
<td>17,049 49,404 15,039</td>
<td>49 204 39</td>
<td>170 492 150</td>
</tr>
</tbody>
</table>
their values, and the remaining reduced-size master problem was solved to integer optimality to find a starting integer-feasible solution for the $D^2$ algorithm. At each $D^2$ iteration, the master problem was solved to integer optimality. When the first-stage solution failed to change from one iteration to the next, all subproblems were solved to integer optimality and a Laporte-Louveaux type cut was added to the master problem. The algorithm terminated when the optimality gap fell below 0.01% ($\epsilon = 0.01\%$ in step 4 of $D^2$) or after 7200 seconds. We ran our experiments using CPLEX 11.1 on a 2.4 GHz processor with 4 GB of memory running Linux.

The disjunction variable for 0-1 cuts was chosen by finding the variable that was fractional in the most number of scenarios. The GUB disjunctions were determined by finding the GUB set $G$ that most frequently contained a fractional variable. For each scenario $\omega$ with a fractional variable in $G$, the starting time of the job $j \in J$ was calculated as $ar{s}(\omega) = \sum_{t=T_0+1}^{T+T_0-p_j+1} t y_{jt}(\omega)$. The $\nu$ value used to partition the GUB set $G$ was chosen as

$$\nu = \lfloor \bar{s}(\omega^*) \rfloor,$$

where

$$\omega^* = \arg\min_{\omega \in \mathcal{F}} \left| \bar{s}(\omega) - \frac{1}{|\mathcal{F}|} \sum_{\omega' \in \mathcal{F}} \bar{s}(\omega') \right|,$$

and $\mathcal{F} \subseteq \Omega$ is the set of scenarios that have fractional elements in GUB set $G$. Selecting $\nu$ in this manner attempts to maximize the number of scenarios in which the GUB disjunction is violated while guaranteeing at least one scenario will violate the disjunction.

### 4.4. Computational Results

Tables 3 and 4 display the results of our computational experiments for $D^2$ with 0-1 cuts ($D^201$), $D^2$ with 0-1 cuts and restricted CGLP ($D^201$-R), $D^2$ with 0-1 cuts and restricted CGLP with strengthening ($D^201$-RS), $D^2$ with GUB cuts ($D^2$GUB), and $D^2$ with GUB cuts and restricted CGLP ($D^2$GUB-R). As explained in §4.2, there are 12 problems labeled A.B.C.D and we generated 5 instances for each problem.

Table 3 displays the minimum, average, and maximum computational time (in seconds) of the five instances for each problem along with the average number of iterations. Problem instances exceeding the time limit are denoted by “$> 7200$” in the “max” column followed by the number of instances out of 5 that were solved within the time limit. For instance, $D^201$ for 5.10.50.1000 had 2 instances that were solved within the time limit.
Table 4 shows the average improvement in computation time of the different $D^2$ implementations compared to $D^201$. The improvement for each instance was calculated as $(D^201_{\text{time}} - \text{other}_{\text{time}})/D^201_{\text{time}}$, where other$_{\text{time}}$ is the computational time of the $D^2$ variant. The improvement reported is the average of the five instances for each problem. Whenever $D^201$ could not solve all 5 instances, we instead report the number of instances solved by the other $D^2$ implementation out of the number of instances that were not solved by $D^201$.

The tables indicate that the restricted CGLP, GUB cuts, and strengthening procedure for 0-1 cuts improve the performance of the $D^2$ algorithm. The restricted CGLP demonstrated significant improvements in computation times. $D^201$-R solved eight instances left unsolved by $D^201$, including all five instances of problem 10.20.50.100, and $D^2$GUB-R solved one more instance than $D^201$-R within the time limit. The computation time for $D^201$-R was about 30% less than $D^201$, and strengthening improved $D^201$-R by about 3% across all 12 problems (60 instances total). The largest improvement over $D^201$ was the variant $D^2$GUB-R, which on average reduced computation time by about 45%.

Although the restricted CGLP is a relaxation of the full-sized CGLP, interestingly, the number of iterations for both $D^201$-R and $D^2$GUB-R are sometimes less than that of their full-sized CGLP counterparts. Balas et al. (1996) experienced a similar phenomenon when comparing performance of full-sized and restricted CGLPs in a disjunctive cutting plane algorithm. They commented that cuts from full-size CGLP may not improve convexification as much as cuts from restricted CGLPs because the full-size CGLP cuts tend to be more parallel to the objective function. As the convexification procedure continues, having cuts that improve convexification in diverse directions is important for the generation of new cuts. Our results agree.

The GUB cuts by themselves were also effective in reducing computation time for the stochastic scheduling problem. $D^2$GUB solved all problem instances solved by $D^201$ plus seven more instances not solved by $D^201$. The average iteration count for $D^2$GUB is less than that of $D^201$ in 7 of 9 problems solved by both implementations, indicating its effectiveness. The improvement in computation times for $D^2$GUB was around 20% on average, omitting problems not solved by $D^201$. As in 0-1 cuts, solving a restricted CGLP significantly reduced computation time. As a result, $D^2$GUB-R achieved on average 45% improvement in computation times.
Table 3: Computation times and average number of iterations for different implementations. No average is reported for problems that had instances not solved within the time limit. In this case, the number in parenthesis indicates the number of instances out of 5 that were solved within the time limit.

<table>
<thead>
<tr>
<th>Problem</th>
<th>$D^201$</th>
<th>$D^201$-R</th>
<th>$D^201$-RS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min avg max iters</td>
<td>min avg max iters</td>
<td>min avg max iters</td>
</tr>
<tr>
<td>5.10.30.10</td>
<td>3 48.3 173 39.8</td>
<td>1 12 24 12</td>
<td>1 10.6 21 9.8</td>
</tr>
<tr>
<td>5.10.30.1000</td>
<td>717 884.6 1339 13.8</td>
<td>711 849 1270 15</td>
<td>655 797 1199 10</td>
</tr>
<tr>
<td>5.10.50.10</td>
<td>32 126.2 263 57</td>
<td>12 65 169 36</td>
<td>11 59 141 29.4</td>
</tr>
<tr>
<td>5.10.50.1000</td>
<td>6781 &gt;7200 (2)</td>
<td>6293 &gt;7200 (3)</td>
<td>6231 &gt;7200 (3)</td>
</tr>
<tr>
<td>10.10.30.10</td>
<td>15 30.6 58 35.5</td>
<td>9 19 32 33</td>
<td>8 18.4 31 30.2</td>
</tr>
<tr>
<td>10.10.30.1000</td>
<td>562 642.6 768 30.6</td>
<td>493 584 601 33</td>
<td>531 593.8 658 33.4</td>
</tr>
<tr>
<td>10.10.50.10</td>
<td>20 102 204 50.4</td>
<td>17 42 120 67</td>
<td>6 46.2 111 54.8</td>
</tr>
<tr>
<td>10.10.50.1000</td>
<td>2314 335.6 455 66.4</td>
<td>2189 2651 3006 80</td>
<td>2211 2505.2 2848 75.6</td>
</tr>
<tr>
<td>10.20.30.10</td>
<td>81 231.4 584 86.2</td>
<td>64 180 390 81</td>
<td>66 171 445 67</td>
</tr>
<tr>
<td>10.20.30.1000</td>
<td>2861 4595.6 615 150.2</td>
<td>2615 3856 5084 135</td>
<td>2631 3844.2 5286 126</td>
</tr>
<tr>
<td>10.20.50.10</td>
<td>1003 &gt;7200 (2)</td>
<td>50 &gt;7200 (4)</td>
<td>150 1687.6 6699 259.4</td>
</tr>
<tr>
<td>10.20.50.1000</td>
<td>&gt;7200 (0)</td>
<td>3731 5426 6921 350</td>
<td>3404 &gt;7200 (4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem</th>
<th>$D^2$GUB</th>
<th>$D^2$GUB-R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min avg max iters</td>
<td>min avg max iters</td>
</tr>
<tr>
<td>5.10.30.10</td>
<td>9 26.6 78 33.8</td>
<td>1 12 21 10</td>
</tr>
<tr>
<td>5.10.30.1000</td>
<td>810 951.6 1396 20.2</td>
<td>635 760 1130 8</td>
</tr>
<tr>
<td>5.10.50.10</td>
<td>22 108.2 226 46.6</td>
<td>13 54 172 45</td>
</tr>
<tr>
<td>5.10.50.1000</td>
<td>6514 &gt;7200 (3)</td>
<td>6111 &gt;7200 (3)</td>
</tr>
<tr>
<td>10.10.30.10</td>
<td>12 21.6 36 27.8</td>
<td>8 15 26 27</td>
</tr>
<tr>
<td>10.10.30.1000</td>
<td>446 493.2 520 29.4</td>
<td>419 446 474 29</td>
</tr>
<tr>
<td>10.10.50.10</td>
<td>16 75.4 198 56.6</td>
<td>8 22 41 35</td>
</tr>
<tr>
<td>10.10.50.1000</td>
<td>1646 1997.6 2253 63</td>
<td>1521 1798 1992 59</td>
</tr>
<tr>
<td>10.20.30.10</td>
<td>81 190.2 447 65</td>
<td>57 146 403 58</td>
</tr>
<tr>
<td>10.20.30.1000</td>
<td>2776 4092.8 5491 118</td>
<td>2632 3854 5172 118</td>
</tr>
<tr>
<td>10.20.50.10</td>
<td>760 3232.8 5876 344.2</td>
<td>253 823 1716 252</td>
</tr>
<tr>
<td>10.20.50.1000</td>
<td>4700 &gt;7200 (3)</td>
<td>2292 3145 4093 269</td>
</tr>
</tbody>
</table>
Table 4: Percentage improvement in computation times over $D^201$. The two numbers in parenthesis indicate the number of instances solved by the modified $D^2$ algorithm out of the number of instances not solved by $D^201$.

<table>
<thead>
<tr>
<th>Problem</th>
<th>$D^201$-R</th>
<th>$D^201$-RS</th>
<th>$D^2$GUB</th>
<th>$D^2$GUB-R</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.10.30.10</td>
<td>74.27%</td>
<td>78.01%</td>
<td>44.81%</td>
<td>75.93%</td>
</tr>
<tr>
<td>5.10.30.1000</td>
<td>4.05%</td>
<td>9.90%</td>
<td>-7.57%</td>
<td>14.09%</td>
</tr>
<tr>
<td>5.10.50.10</td>
<td>48.18%</td>
<td>53.25%</td>
<td>14.26%</td>
<td>56.89%</td>
</tr>
<tr>
<td>5.10.50.1000</td>
<td>(1/3)</td>
<td>(1/3)</td>
<td>(1/3)</td>
<td>(1/3)</td>
</tr>
<tr>
<td>10.10.30.10</td>
<td>39.22%</td>
<td>39.87%</td>
<td>29.41%</td>
<td>57.52%</td>
</tr>
<tr>
<td>10.10.30.1000</td>
<td>9.12%</td>
<td>7.57%</td>
<td>23.25%</td>
<td>30.66%</td>
</tr>
<tr>
<td>10.10.50.10</td>
<td>58.82%</td>
<td>54.71%</td>
<td>26.08%</td>
<td>78.82%</td>
</tr>
<tr>
<td>10.10.50.1000</td>
<td>20.51%</td>
<td>24.90%</td>
<td>40.11%</td>
<td>46.10%</td>
</tr>
<tr>
<td>10.20.30.10</td>
<td>22.10%</td>
<td>26.10%</td>
<td>17.80%</td>
<td>36.91%</td>
</tr>
<tr>
<td>10.20.30.100</td>
<td>16.09%</td>
<td>16.35%</td>
<td>10.94%</td>
<td>16.13%</td>
</tr>
<tr>
<td>10.20.50.10</td>
<td>(2/3)</td>
<td>(3/3)</td>
<td>(3/3)</td>
<td>(3/3)</td>
</tr>
<tr>
<td>10.20.50.1000</td>
<td>(5/5)</td>
<td>(4/5)</td>
<td>(3/5)</td>
<td>(5/5)</td>
</tr>
</tbody>
</table>

$D^2$GUB-R with strengthening was not effective. Strengthening can only be applied when $\lambda_{02} + \lambda_{12} \neq 0$, and this condition was rarely achieved during GUB cut generation. Hence, strengthening was rarely possible and computational results were basically the same as $D^2$GUB-R without cut strengthening.

4.5. Further Analysis

The results in Table 4 show that computational improvement appears to decrease as $|\Omega|$, the number of realizations of the random vector $\tilde{\omega}$, increases. To gain a better understanding of this, we solved problem 10.10.30 with $|\Omega| = 10, 100, 200, \ldots, 1000$ using $D^201$, $D^201$-RS, and $D^2$GUB-R. Figure 1 shows the improvement in total computation time and cut generation time per $D^2$ iteration for $D^201$-RS and $D^2$GUB-R over $D^201$. The first point, $|\Omega| = 10$, and the last point, $|\Omega| = 1000$, in Figure 1a are reported in Table 4. From $|\Omega| = 10$ to 100, for both $D^2$ variants, there is a drop in improvement in total time and improvement in cut generation time per $D^2$ iteration. The improvement of $D^2$GUB-R appears to stabilize for $|\Omega| \geq 100$ while the improvement of $D^201$-RS experiences a slightly negative trend.

Figure 2 shows the absolute amount of time spent per $D^2$ iteration on cut generation, right-hand-side convexification, and all other computations. Time spent per $D^2$ iteration on the RHSLP and other computations were similar for the three algorithms. However, time spent per $D^2$ iteration on CGLP was notably different. $D^201$ spent the most amount of time per $D^2$ iteration on cut generation, $D^201$-RS spent a little less time, and $D^2$GUB-R spent
the least amount of time per $D^2$ iteration. The graph suggests reduction in cut generation time per $D^2$ iteration is the major factor in reduction of total computational time.

The difference in cut generation time per $D^2$ iteration appears to be due to the number of L-shaped iterations required to generate the cut. We noticed that $D^2$01-RS required around 5 L-shaped iterations to solve the CGLP for $|\Omega| = 10$. As $|\Omega|$ increased, the number of CGLP L-shaped iterations rose to around 15-20 with an occasional CGLP requiring more. CGLP L-shaped iterations for $D^2$GUB-R usually remained around 5 regardless of $|\Omega|$. This explains why in Figure 2 cut generation time per $D^2$ iteration for $D^2$01-RS is around 4 times more than cut generation per $D^2$ iteration for $D^2$GUB-R CGLP.

The fewer number of CGLP L-shaped iterations required for GUB disjunctions may be due to two factors. First, $\lambda_{02}$ and $\lambda_{12}$ appear in more constraints in the CGLP for GUB
disjunctions than in the CGLP for 0-1 disjunctions since GUB disjunctions affect all variables in GUB set $G$ whereas 0-1 disjunctions affect only one variable. The extra constraints may provide more information to the CGLP master problem on good values of $\lambda_{02}$ and $\lambda_{12}$ so that less feedback is required from the CGLP second-stage. Second, $\lambda_{12}$ is not present in the CGLP second-stage (5h) for GUB disjunctions, reducing the likelihood of the second-stage being infeasible for a set of $\lambda$’s passed from the CGLP master problem. To test this explanation, we implemented an alternative GUB disjunction

$$\left(- \sum_{j \in G_0} y_j \geq 0 \right) \cup \left( \sum_{j \in G_0} y_j \geq 1 \right),$$

which we denote as the GUB01 disjunction. This is equivalent to the GUB disjunction in (9); however, the GUB01 disjunction in (23) results in $\lambda_{12}$ appearing in the CGLP second-stage, whereas $\lambda_{12}$ is absent in the CGLP second-stage for GUB disjunctions. Figure 3 displays the cut generation time per $D^2$ iteration reported in Figure 2c along with cut generation time per $D^2$ iteration for $D^2$ with GUB01 disjunctions and restricted CGLP ($D^2$GUB01-R). The figure shows that $D^2$GUB01-R takes less cut generation time per $D^2$ iteration than $D^2$01 and $D^2$01-R but more time than $D^2$GUB-R. The $D^2$GUB01-R CGLP required 5-10 L-shaped iterations with an occasional CGLP requiring more. The $D^2$GUB01-R results support the ideas that: (1) the CGLP master problem for GUB disjunctions provides more information on good values of $\lambda_{02}$ and $\lambda_{12}$ than 0-1 disjunctions, and (2) the presence of $\lambda_{12}$ in the CGLP second-stage affects the number of iterations required to generate a cut. We note that total computational time for $D^2$GUB01-R was approximately 20% larger than $D^2$GUB-R.

Finally, we tested the effect of parameter $\rho$ on the performance of the enhancements. Recall that $\rho$ controls the ratio of resources consumed to resources available. When $\rho < 1$, there are more resources available than needed; $\rho > 1$ indicates scarcity of resources. For this test, we created three instances of problem 10.20.30.100 for each value of $\rho \in [0.7, 0.9, 1.1, 1.3]$. We solved these instances using $D^2$01, $D^2$01-RS and $D^2$GUB-R. As before, $D^2$GUB-R outperformed the other implementations. Our results indicate that the percentage improvement due to enhancements slightly increases as $\rho$ increases. This relationship is, however, quite weak. Our results also suggest that $\rho$ is not a good indicator of instance difficulty for this scheduling problem.
5. Conclusions

In this paper, we explored the use of generating cuts in a subspace of subproblem variables and alternative disjunctions based on the GUB structure in the second-stage within the \( D^2 \) algorithm. We introduced a scheduling problem with uncertain number of jobs and presented computational experiments on instances of this problem. Our computational tests indicate that the restricted CGLP greatly speeds up the \( D^2 \) algorithm although the improvement for 0-1 cuts decreases as the number of scenarios increases. We also found that cut strengthening can be beneficial for 0-1 disjunctions with minimal additional computational burden.

The alternative disjunctions exploit the special GUB structure found in the second-stage of many SMIPs. The GUB disjunctive cuts seem to have less computational burden than the 0-1 cuts and our computational experiments indicate the GUB cuts are more effective for the stochastic scheduling problem studied in the paper. Furthermore, the effectiveness of GUB cuts seem to be fairly stable as the number of realizations of \( \tilde{\omega} \) increases. \( D^2 \) with GUB cuts using the restricted CGLP resulted in the most improvement, cutting computational time on average by 45% over the base algorithm, \( D^201 \). Future work include extended computational studies on other SMIPs with GUB constraints but with other characteristics than the stochastic scheduling problem studied in this paper (e.g., model of Zhang et al. (2010)) as well testing the effectiveness of the enhancements on problem- and data-specific features of these problems. Parallel implementations can further reduce computation time.

Another area of future research is applying the restricted problems to the RHSLP convexification process within \( D^2 \) and \( D^2\text{-BAC} \). Perregaard and Balas (2001) explore generating cuts from multiple 0-1 disjunctions. Recall that the nonzeros in the CGLP are proportional to
the number of disjunctions resulting in prohibitively large CGLPs for even a small number of disjunctions. Perregaard and Balas overcome this difficulty by decomposing the CGLP based on disjunctions (whereas we decompose based on scenarios) and found favorable results. The disjunctions are chosen corresponding to nodes in a partially solved branch-and-bound tree. In the SMIP setting, $D^2$-BAC uses disjunctions from a partially solved branch-and-bound tree to convexify the recourse function. Future $D^2$-BAC research could investigate combining multiple 0-1 disjunctions for set convexification formed from the partially solved tree along with recourse function convexification from the same tree.

**Acknowledgments**

The authors thank Suvrajeet Sen for valuable discussions and two anonymous referees for helpful comments that significantly improved the presentation of the paper. This research is supported in part by the National Science Foundation under Grant EFRI-0835930.

**References**


30