Noname manuscript No.
(will be inserted by the editor)

A Combinatorial Cut-and-Lift Procedure with an Application to 0-1 Chance Constraints

Margarita P. Castro · Andre A. Cire · J. Christopher Beck

Received: date / Accepted: date

Abstract Cut generation and lifting are key components for the performance of state-of-the-art mathematical programming solvers. This work proposes a new general cut-and-lift procedure that exploits the combinatorial structure of 0-1 problems via a binary decision diagram (BDD) encoding of their constraints. We present a general framework that can be applied to a large range of binary optimization problems and show its applicability for normally distributed chance constraints. We identify conditions for which our lifted inequalities are facet-defining and derive a new BDD-based cut generation linear program. Such a model serves as a basis for a max-cut combinatorial algorithm over the BDD that can be applied to derive valid cuts more efficiently. Our numerical results show encouraging performance when incorporated into a state-of-the-art mathematical programming solver, significantly reducing the root node gap, increasing the number of problems solved, and reducing the run-time by a factor of three on average.

Keywords Lifting · Cutting Planes · Decision Diagrams · Binary Optimization

Margarita P. Castro
Department of Mechanical and Industrial Engineering, University of Toronto
E-mail: mpcastro@mie.utoronto.ca

Andre A. Cire
Dept. of Management, University of Toronto Scarborough and
Rotman School of Management
E-mail: andre.cire@rotman.utoronto.ca

J. Christopher Beck
Department of Mechanical and Industrial Engineering, University of Toronto,
E-mail: jcb@mie.utoronto.ca

1 Introduction

Cutting plane methodologies have played a key role in the theoretical and computational development of mathematical programming [19,43]. Extensive literature has focused on cuts that exploit special problem substructure, leading to an array of techniques that are now integral into state-of-the-art solvers [37]. For general problems, cuts are obtained either by leveraging disjunctive reformulations [10,11] or by *lifting*, i.e., relaxing an initial inequality so that it is valid for a higher-dimensional polyhedron [29,39,54].

In this paper, we study both a cut generation procedure and a lifting approach for general binary optimization problems of the form

$$\max_{\boldsymbol{x} \in X \subseteq \{0,1\}^n} \boldsymbol{c}^\top \boldsymbol{x}, \tag{BP}$$

where the feasible set X is arbitrary, e.g., possibly represented by a conjunction of linear and/or non-linear constraints. Our methodologies consist of exploiting *network structure* via a binary decision diagram (BDD) embedding of X. A BDD is a graphical model that represents solutions as paths in a directed acyclic graph, which in our context can be viewed as a network-flow reformulation of X. Such a model is potentially orders of magnitude smaller than an explicit representation of X as it identifies and merges equivalent partial solutions. Several BDD encodings have already been investigated for linear and non-linear problems [13,14,40] and are used to exploit submodularity [15] or more general combinatorial structure [17].

We propose a sequential lifting procedure that can be applied to any initial inequality (e.g., given by another cutting-plane technique). The lifting algorithm uses 0-1 disjunctions derived from a BDD representation of X to rotate inequalities while maintaining their validity. We show that each step of our sequential lifting, when applicable, increases the dimension of the face by at least one, and we establish conditions for which the inequality becomes facet-defining. We also draw connections between our procedure and existing lifting techniques from disjunctive programming [9], showing that our approach generalizes well-known lifting procedures for 0-1 inequalities [8,31,48].

For our cut generation approach, we propose a new linear formulation of the BDD polytope based on capacitated flows, which leads to an alternative cut generation linear program (CGLP) for separating infeasible points from X. We show that the set of cuts derived from this model defines the convex hull of the solutions encoded by the BDD, i.e., X. Moreover, in contrast to recent cutting-plane algorithms based on BDDs [28,51], our CGLP does not require any additional information about X, such as interior points or normalization constraints. Finally, for practical purposes, we build on this model to present a weaker but computationally faster alternative that solves a combinatorial max-flow/min-cut problem over the BDD to generate valid inequalities.

For optimization problems where a BDD for X may be exponentially large in n, our lifting and cut procedures remain valid when considering instead a limited-size *relaxed* BDD for BP, i.e., where the BDD encodes a superset of X.

Several efficient methods exist to build relaxed BDDs, such as only considering a subset of the problem constraints [17]. This approach is similar in spirit, e.g., to when a linear relaxation is used to lift cover inequalities of a single knapsack constraint [8]. Nonetheless, here we exploit the discrete relaxation provided by the BDD as opposed to a continuous relaxation, which captures some of the combinatorial structure of the problem.

As a case study, we apply our combinatorial cut-and-lift procedure to a class of 0-1 chance-constrained problems. Chance constraints are common in stochastic optimization to model uncertainty or enforce robustness [5,27,50, 25]. Existing solutions methods include sampling [41,46], scenario reformulation [2,42], and Lagrangian relaxation [1], for which both the lifting and cutting-planes presented here can be useful. In particular, we focus on binary problems with normally distributed chance constraints:

$$\max_{\boldsymbol{x}\in\{0,1\}^n} \quad \left\{\boldsymbol{c}^\top \boldsymbol{x} : \quad \mathbb{P}(\boldsymbol{a}_j^\top \boldsymbol{x} \leq b_j) \geq \epsilon_j, \forall j \in \{1,...,m\}\right\}, \tag{CC}$$

where, for each j, $\boldsymbol{a}_j \in \mathbb{R}^n$ is a vector of random variables with a joint normal distribution and ϵ_j is a threshold probability. Each inequality of CC can be rewritten as a second-order cone (SOC) constraint [47], which are amenable to commercial solvers such as CPLEX [34] and Gurobi [30]. However, solver performance on problems with SOC constraints is still not comparable to the integer linear programming (ILP) case, despite advances in linearization methods [52,53] and cutting-planes for SOC [6,7,38].

We investigate problems with multiple SOC inequalities, each reformulated with an appropriate BDD encoding. We experiment on the knapsack chance-constraint benchmark [6,35] and over 270 randomly generated instances with joint-distributed chance constraints, incorporating both our general cutting and lifting approaches into CPLEX. We show that our combinatorial cut-and-lift procedure achieves a 52.2% average root gap reduction, has comparable average solving time, and solves 17 more instances on the knapsack benchmark when compared to existing cut-and-lift methodologies [6]. Similarly, our procedure outperforms CPLEX on the random dataset by achieving a 35.3% average root gap reduction, solving 31 more instances (168 vs. 137), and reducing the mean run-time threefold.

The remainder of the paper is as follows. §2 describes related works in the BDD and lifting literature. §3 introduces notation and background material. §4 describes our combinatorial lifting procedure while §5 details our BDD-based cutting-plane algorithm. §6 introduces the chance-constraint problem and describes the BDD encoding for SOC inequalities. Lastly, §7 and §8 present the empirical evaluation and final remarks, respectively.

2 Related Work

Recent research has shown the versatility of BDDs for modeling linear and non-linear inequalities [4,32,18,15] and there is a growing literature on BDD

encodings for vehicle routing [23,49,36], scheduling [26,20,21,33], and other combinatorial optimization problems [14,16,24,17]. Within the context of this work, Becker et al. (2005) [12] presented the first BDD cut generation procedure based on an iterative subgradient algorithm that relies on a longest-path problem over the BDD. Behle (2007) [13] formalized this procedure and proposed a branch-and-cut algorithm that employs BDDs to generate exclusion and implication cuts. The author also introduced the network flow model employed by most BDD cutting-plane procedures [28,40,51].

Tjandraatmadja and van Hoeve (2019) [51] recently demonstrated how to generate target cuts from polar sets, using relaxed BDDs to obtain more computationally tractable procedures. Davarnia and van Hoeve (2020) [28] proposed an iterative method to generate outer-approximations for non-linear inequalities. Both works introduce BDD-based cutting models, which we further discuss and compare to our approach in §5.4. Lastly, Lozano and Smith (2018) [40] designed a new class of BDD cuts for two-stage stochastic programming problems using BDDs to encode second-stage decisions. The authors propose a CGLP where arc capacities are given by first-stage decision, which resembles our approach (see §5).

While the literature on BDD cutting-plane procedures has recently grown, to the best of our knowledge, this is the first work that leverages BDDs for lifting existing inequalities. Our combinatorial lifting, however, has a strong relationship to lifting algorithms based on 0-1 disjunctions [9], including procedures for knapsack inequalities [8,45,44] and submodular functions [31,6]. In particular, our methodology is closely related to the n-step lifting procedure by Perregaard and Balas (2001) [48], which generalizes previous works. A brief description of this procedure can be found in §3 and its relationship to our combinatorial lifting algorithm is delineated in §4.4.

3 Background

This section introduces the notation used throughout this work and the background material on decision diagrams for optimization. We also discuss previous concepts of disjunctive programming that are related to our methodology.

For convenience, we assume $n \geq 1$ and let $I := \{1, ..., n\}$ represent the component indices of any point $\boldsymbol{x}$ in an n-dimensional set.

Facets and Convex Hulls. We denote by $\dim(P)$ the dimension of a polytope $P \subseteq [0,1]^n$. An inequality $\boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0$ with $\boldsymbol{\pi} \in \mathbb{R}^n$ and $\pi_0 \in \mathbb{R}$ is valid for P if $\boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0$ holds for all $\boldsymbol{x} \in P$. The inequality defines a face of P if $F(\boldsymbol{\pi}) := \{\boldsymbol{x} \in P : \boldsymbol{\pi}^\top \boldsymbol{x} = \pi_0\}$ is not empty, i.e., the inequality *supports* P. A face $F(\boldsymbol{\pi})$ is a *facet* if $\dim(F(\boldsymbol{\pi})) = \dim(P) - 1$; in such a case, $\boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0$ is *facet-defining*. Finally, we denote the convex hull of P by $\text{conv}(P)$.

Binary Decision Diagrams. A BDD $\mathcal{B}$ is an extended representation of a set $X_{\mathcal{B}} \subseteq \{0,1\}^n$ as a network. Specifically, $\mathcal{B} = (\mathcal{N}, \mathcal{A})$ is a layered directed acyclic graph with node set $\mathcal{N}$ and arc set $\mathcal{A}$. The node set $\mathcal{N}$ is partitioned

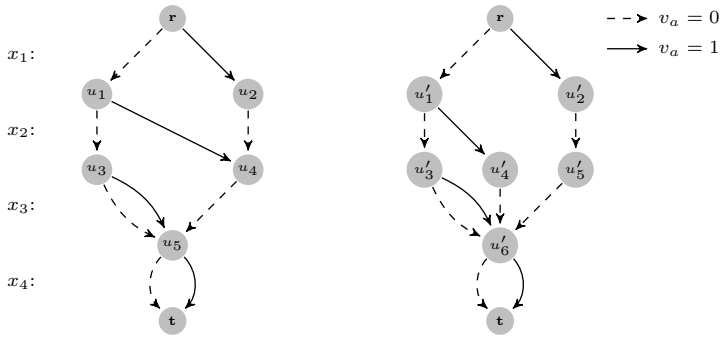

Fig. 1 Two BDDs $\mathcal{B}_1$ (left-hand side) and $\mathcal{B}_2$ (right-hand side) with $X_{\mathcal{B}_1} = X_{\mathcal{B}_2} = \{\boldsymbol{x} \in \{0,1\}^4 : 7x_1 + 5x_2 + 4x_3 + x_4 \leq 8\}$. $\mathcal{B}_1$ is reduced.

into $n+1$ *layers* $\mathcal{N} = (\mathcal{N}_1, ..., \mathcal{N}_{n+1})$. The first and last layers are the singletons $\mathcal{N}_1 = \{\mathbf{r}\}$ and $\mathcal{N}_{n+1} = \{\mathbf{t}\}$, respectively, where $\mathbf{r}$ is the root node and $\mathbf{t}$ is the terminal node. An arc $a = (u, v) \in \mathcal{A}$ has a source node $s(a) = u$ and a target node $t(a) = v$ in consecutive layers, i.e., $v \in \mathcal{N}_{i+1}$ whenever $u \in \mathcal{N}_i$ for $i \in I$.

The points of $X_{\mathcal{B}}$ are mapped to paths in the network, as follows. With each arc $a \in \mathcal{A}$ we associate a value $v_a \in \{0, 1\}$, where a node $u \in \mathcal{N}$ has at most one arc of each value emanating from it. Given an arc-specified $\mathbf{r}-\mathbf{t}$ path $p = (a_1, ..., a_n)$ with $s(a_1) = \mathbf{r}$ and $t(a_n) = \mathbf{t}$, we let $\boldsymbol{x}^p := (v_{a_1}, v_{a_2}, \ldots, v_{a_n}) \in \{0, 1\}^n$ be the n-dimensional point encoded by path p. Thus, if $\mathcal{P}$ is the set of all $\mathbf{r}-\mathbf{t}$ paths in $\mathcal{B}$, the set of points represented by the BDD is $X_{\mathcal{B}} = \bigcup_{p \in \mathcal{P}} \{\boldsymbol{x}^p\}$.

A BDD $\mathcal{B}$ is *exact* for set $X \subseteq \{0, 1\}^n$ when $X = X_{\mathcal{B}}$, i.e., there is a one-to-one relationship between the points in X and the $\mathbf{r} - \mathbf{t}$ paths in $\mathcal{B}$. Alternatively, $\mathcal{B}$ is *relaxed* when $X \subseteq X_{\mathcal{B}}$, i.e., every point in X maps to a path in $\mathcal{B}$ but the converse is not necessarily true.

Example 1 Consider $X = \{\boldsymbol{x} \in \{0, 1\}^4 : 7x_1 + 5x_2 + 4x_3 + x_4 \leq 8\}$. Figure 1 illustrates two exact BDDs for X: $\mathcal{B}_1$ on the left-hand side and $\mathcal{B}_2$ on the right-hand side. Dashed and solid arcs have a value of 0 and 1, respectively. Each point $\boldsymbol{x} \in X$ is represented by a path in $\mathcal{B}_1$ and $\mathcal{B}_2$. For example, $\boldsymbol{x} = (1, 0, 0, 1) \in X$ is encoded by the path $((\mathbf{r}, u_2), (u_2, u_4), (u_4, u_5), (u_5, \mathbf{t}))$ in $\mathcal{B}_1$, and by the path $((\mathbf{r}, u'_2), (u'_2, u'_5), (u'_5, u'_6), (u'_6, \mathbf{t}))$ in $\mathcal{B}_2$. □

A BDD $\mathcal{B}$ is *reduced* if it is the smallest network (with respect to number of nodes) that represents the set $X_{\mathcal{B}}$. There exists a unique reduced BDD for a given ordering of the indices I. Furthermore, given any $\mathcal{B}$ and an ordering, we can obtain its associated reduced BDD in polynomial time in the size of $\mathcal{B}$ [22]. For instance, $\mathcal{B}_1$ in Figure 1 is reduced and can be obtained by merging nodes u'_4 and u'_5 from $\mathcal{B}_2$ and adjusting their emanating arcs appropriately.

Several exact and relaxed BDD construction mechanisms are available for general and specialized discrete optimization problems [17,15,51]. These techniques are based on either reformulating the problem as a dynamic program, where $\mathcal{B}$ represents an underlying state-transition graph, or by separating in-

feasible paths of a relaxed BDD. It is often the case that BDDs can be exponentially smaller than enumerating $X_{\mathcal{B}}$ explicitly [17]. If exact BDDs are too large, relaxed BDDs can be built for X by either imposing a limit on the number of nodes (e.g., a polynomial in the problem input) or by considering only a subset of the problem constraints. We discuss the construction and relaxation techniques used for our chance-constraint case study in §6.

An Iterative Lifting Procedure based on Disjunctive Programming. We now describe Perregaard and Balas (2001) [48] n-step lifting procedure that is closely related to our methodology. Given a mixed-integer linear programming (MILP) problem of the form $\max_{\boldsymbol{x}}\{\boldsymbol{c}^\top \boldsymbol{x} \colon A\boldsymbol{x} \leq \boldsymbol{b}, x_i \in \mathbb{Z}\ \forall i \in I' \subseteq I\}$, the authors propose the relaxation

$$\max_{\boldsymbol{x}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} :\ A\boldsymbol{x} \leq \boldsymbol{b},\ \bigvee_{k \in K} D^k \boldsymbol{x} \leq \boldsymbol{d}^k, x_i \in \mathbb{Z} \quad \forall i \in I'' \subset I' \right\}, \tag{DP}$$

where fewer variables are constrained to be integral. The set K that defines the disjunctive constraints is typically derived by considering the 0-1 integrality constraints of individual variables (e.g., $x_i \leq 0 \vee x_i \geq 1$).

Let P_{DP} be the set of solutions of DP. The n-step procedure considers two inputs: (a) an inequality $\boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0$ that supports $\text{conv}(P_{DP})$; and (b) an arbitrary target inequality $\widetilde{\boldsymbol{\pi}}^\top \boldsymbol{x} \leq \widetilde{\pi}_0$ that is tight for all integer points in $F(\boldsymbol{\pi})$. The procedure uses a parameter γ to rotate the supporting inequality towards the target inequality, generating a new lifted inequality $(\boldsymbol{\pi}+\gamma\widetilde{\boldsymbol{\pi}})^\top \boldsymbol{x} \leq \pi_0+\gamma\widetilde{\pi}_0$ that is valid for $\text{conv}(P_{DP})$. In particular, if $\widetilde{\boldsymbol{\pi}}^\top \boldsymbol{x} \leq \widetilde{\pi}_0$ is *not* valid for P_{DP}, it can be shown that there is a finite maximal γ given by the disjunctive program

$$\begin{aligned}
\gamma^* = \min_{\boldsymbol{x}, x_0} \quad & \pi_0 x_0 - \boldsymbol{\pi}^\top \boldsymbol{x} \\
\text{s.t.} \quad & A\boldsymbol{x} - \boldsymbol{b}x_0 \leq 0, \\
& \bigvee_{k \in K} D^k \boldsymbol{x} - \boldsymbol{d}^k x_0 \leq 0, \\
& \widetilde{\pi}_0 x_0 - \widetilde{\boldsymbol{\pi}}^\top \boldsymbol{x} = -1, \\
& x_0 \geq 0, x_i \in \mathbb{Z}\ \forall i \in I'' \subset I'.
\end{aligned}$$

Under the same assumptions, the lifted inequality becomes a facet of $\text{conv}(P_{DP})$ if the procedure is repeated n times, using the rotated inequality and an appropriate target inequality.

Similarly, our approach is a sequential procedure that relies on disjunctive inequalities for lifting. It differs from existing methods in that we exploit the combinatorial structure encoded by a BDD as opposed to a disjunctive program relaxation. Such a BDD may encode, e.g., complex non-linear constraints that are not necessarily convex [15]. Furthermore, we also exploit the network to derive a tractable and efficient way to compute several disjunctions simultaneously, while previous algorithms are typically restricted to a small number of disjunctions [48]. To highlight the connection between methods, we show in §4.4 that our lifting technique becomes a special case of Perregaard and Balas (2001) under a restricted setting.

4 Combinatorial Lifting

We now present our combinatorial lifting procedure and develop its structural properties. Throughout this section, we assume that, for a given $X \subseteq \{0,1\}^n$,

(a) $\boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0$ is a valid inequality that supports $\text{conv}(X)$;
(b) $\mathcal{B}$ is an exact BDD for X, i.e., $X_{\mathcal{B}} = X$; and
(c) For any $i \in I$, there exists $\boldsymbol{x}, \boldsymbol{x}' \in X$ such that $x_i = 0$ and $x'_i = 1$.

Assumption (a) is a common lifting condition that is satisfied by setting $\pi_0 := \max_{\boldsymbol{x} \in X} \{\boldsymbol{\pi}^\top \boldsymbol{x}\}$. This, in turn, can be enforced in linear time in the size of $\mathcal{B}$ (see §4.2). Assumption (b) is needed for our theoretical results but it can be relaxed in practice (see §7). For (c), we can soundly remove any i-th component not satisfying the assumption, adjusting n accordingly.

Our goal is to lift $\boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0$ and better represent $\text{conv}(X)$ by exploiting the network structure of $\mathcal{B}$. The resulting cuts are valid for any subset $X' \subseteq X$; e.g., when $\mathcal{B}$ (and hence X) is a relaxation of some feasible set.

We begin in §4.1 by introducing our lifting mechanism based on variable disjunctions. We then present in §4.2 a methodology that computes such a lifting in polynomial time in the size of $\mathcal{B}$ (i.e., in the number of nodes and arcs). Next, in §4.3 we incorporate the technique in a sequential procedure and investigate the dimension of the resulting face. Finally, we depict the relationship with previous disjunctive methodologies in §4.4.

4.1 Disjunctive Slack Lifting

The core element of our lifting procedure is what we denote by *disjunctive slack vector* (or *d-slack* in short). The i-th component of the d-slack indicates the change in the maximum values of the left-hand side of $\boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0$ when varying x_i. This is formalized in Definition 1.

Definition 1 The disjunctive slack vector $\boldsymbol{\lambda}(\boldsymbol{\pi})$ with respect to $\boldsymbol{\pi}$ is

$$\lambda_i(\boldsymbol{\pi}) := \lambda_i^0(\boldsymbol{\pi}) - \lambda_i^1(\boldsymbol{\pi}), \quad \forall i \in I,$$

with $\lambda_i^0(\boldsymbol{\pi}) := \max_{\boldsymbol{x} \in X} \{\boldsymbol{\pi}^\top \boldsymbol{x} : x_i = 0\}$ and $\lambda_i^1(\boldsymbol{\pi}) := \max_{\boldsymbol{x} \in X} \{\boldsymbol{\pi}^\top \boldsymbol{x} : x_i = 1\}$.

For notational convenience, we let $S^-(\boldsymbol{\pi}) := \{i \in I : \lambda_i(\boldsymbol{\pi}) < 0\}$, $S^0(\boldsymbol{\pi}) := \{i \in I : \lambda_i(\boldsymbol{\pi}) = 0\}$, and $S^+(\boldsymbol{\pi}) := \{i \in I : \lambda_i(\boldsymbol{\pi}) > 0\}$ be a partition of I with respect to negative, zero, and positive d-slacks, respectively. Lemma 1 presents key properties of d-slacks used for our main results.

Lemma 1 *For any $\boldsymbol{\lambda}(\boldsymbol{\pi})$ and index $i \in I$,*

(1) $i \in S^-(\boldsymbol{\pi})$ if and only if $x_i = 1$ for all $\boldsymbol{x} \in F(\boldsymbol{\pi})$.
(2) $i \in S^+(\boldsymbol{\pi})$ if and only if $x_i = 0$ for all $\boldsymbol{x} \in F(\boldsymbol{\pi})$.
(3) $i \in S^0(\boldsymbol{\pi})$ if and only if there exists $\boldsymbol{x}, \boldsymbol{x}' \in F(\boldsymbol{\pi})$ with $x_i = 0$ and $x'_i = 1$.

Proof For the necessary conditions, consider first $x_i = 1$ for all $\boldsymbol{x} \in F(\boldsymbol{\pi})$. Since the solutions when optimizing over $\boldsymbol{\pi}^\top \boldsymbol{x}$ must belong to the face $F(\boldsymbol{\pi})$, we must necessarily have $\lambda_i^1(\boldsymbol{\pi}) < \lambda_i^0(\boldsymbol{\pi})$ and the d-slack $\lambda_i(\boldsymbol{\pi})$ is negative. An analogous reasoning holds for the other two cases.

For the sufficient conditions, consider first $i \in S^-(\boldsymbol{\pi})$. Then $\lambda_i^1(\boldsymbol{\pi}) = \pi_0$ and $\lambda_i^0(\boldsymbol{\pi}) < \pi_0$, i.e., all $\boldsymbol{x} \in F(\boldsymbol{\pi})$ are such that $x_i = 1$. The same argument can be applied to the case $i \in S^+(\boldsymbol{\pi})$. Lastly, if $k \in S^0(\boldsymbol{\pi})$, $\lambda_i^0(\boldsymbol{\pi}) = \lambda_i^1(\boldsymbol{\pi}) = \pi_0$. Thus, there exists $\boldsymbol{x} \in F(\boldsymbol{\pi})$ that maximizes $\lambda_i^1(\boldsymbol{\pi})$ (i.e., $x_i = 1$) and $\boldsymbol{x}' \in F(\boldsymbol{\pi})$ that maximizes $\lambda_i^0(\boldsymbol{\pi})$ (i.e., $x'_i = 0$). ■

We now show in Theorem 1 how to apply the d-slacks to lift $\boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0$. In particular, the resulting inequality is valid for X (and thereby $\text{conv}(X)$), the dimension of the face necessarily increases, and points separated by the original inequality are still separated after lifting. This last characteristic is important, e.g., if the input inequality $\boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0$ was derived to separate a fractional point. Note that we require a d-slack with a non-zero component to rotate the inequality, as we later illustrate in Example 2.

Theorem 1 *Suppose $\lambda_i(\boldsymbol{\pi}) \neq 0$ for some $i \in I$. Let $\langle \boldsymbol{\pi}', \pi'_0 \rangle$ be such that*

$$\pi'_j := \begin{cases} \pi_j & \text{if } j \neq i, \\ \pi_j + \lambda_j(\boldsymbol{\pi}) & \text{otherwise,} \end{cases} \qquad \forall j \in I,$$

and

$$\pi'_0 := \begin{cases} \pi_0 & \text{if } i \in S^+(\boldsymbol{\pi}), \\ \pi_0 + \lambda_i(\boldsymbol{\pi}) & \text{otherwise.} \end{cases}$$

The following properties hold:

(1) $\boldsymbol{\pi}'^\top \boldsymbol{x} \leq \pi'_0$ is valid for X.
(2) $F(\boldsymbol{\pi}) \subset F(\boldsymbol{\pi}')$ and $\dim(F(\boldsymbol{\pi}')) \geq \dim(F(\boldsymbol{\pi})) + 1$.
(3) For any $\bar{\boldsymbol{x}} \in [0,1]^n$ with $\boldsymbol{\pi}^\top \bar{\boldsymbol{x}} > \pi_0$, we have that $\boldsymbol{\pi}'^\top \bar{\boldsymbol{x}} > \pi'_0$.

Proof Let $\boldsymbol{x} \in X$. We begin by showing (1) and (2). Assume first that $i \in S^+(\boldsymbol{\pi})$. By construction, $\boldsymbol{\pi}'^\top \boldsymbol{x} \leq \pi'_0 \iff \boldsymbol{\pi}^\top \boldsymbol{x} + \lambda_i(\boldsymbol{\pi}) x_i \leq \pi_0$.

If $x_i = 0$, the lifted inequality is equivalent to the original and therefore valid. Otherwise, if $x_i = 1$, $i \in S^+(\boldsymbol{\pi})$ implies that $\lambda_i(\boldsymbol{\pi}) = \pi_0 - \lambda_i^1(\boldsymbol{\pi})$. Thus,

$$\boldsymbol{\pi}'^\top \boldsymbol{x} \leq \pi'_0 \iff \boldsymbol{\pi}^\top \boldsymbol{x} + \pi_0 - \lambda_i^1(\boldsymbol{\pi}) \leq \pi_0 \iff \boldsymbol{\pi}^\top \boldsymbol{x} \leq \lambda_i^1(\boldsymbol{\pi}).$$

The last inequality above holds because we are restricting to the case $x_i = 1$ and, by definition, $\lambda_i^1(\boldsymbol{\pi}) = \max_{\boldsymbol{x}' \in X} \{\boldsymbol{\pi}^\top \boldsymbol{x}' : x'_i = 1\}$. Since $x'_i = 0$ for all $\boldsymbol{x}' \in F(\boldsymbol{\pi})$ (Lemma 1), the lifted inequality is tight for all $\boldsymbol{x}' \in F(\boldsymbol{\pi})$, i.e., $F(\boldsymbol{\pi}) \subset F(\boldsymbol{\pi}')$. Notice also that this inequality is tight for $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}' \in X} \{\boldsymbol{\pi}^\top \boldsymbol{x}' : x'_i = 1\}$, i.e., $\boldsymbol{x}^* \in F(\boldsymbol{\pi}')$. Then, $\boldsymbol{x}^*$ is affinely independent to all points of $F(\boldsymbol{\pi})$ and therefore $\dim(F(\boldsymbol{\pi}')) \geq \dim(F(\boldsymbol{\pi})) + 1$.

Assume now that $i \in S^-(\boldsymbol{\pi})$. Once again by construction,

$$\boldsymbol{\pi}'^\top \boldsymbol{x} \leq \pi'_0 \iff \boldsymbol{\pi}^\top \boldsymbol{x} + \lambda_i(\boldsymbol{\pi}) x_i \leq \pi_0 + \lambda_i(\boldsymbol{\pi}).$$

If $x_i = 1$, the lifted inequality is equivalent to the original and therefore valid. Otherwise, if $x_i = 0$, $i \in S^-(\boldsymbol{\pi})$ implies that $\lambda_i(\boldsymbol{\pi}) = \lambda_i^0(\boldsymbol{\pi}) - \pi_0$. Thus,

$$\boldsymbol{\pi}'^\top \boldsymbol{x} \leq \pi_0' \iff \boldsymbol{\pi}^\top \boldsymbol{x} \leq \pi_0 + \lambda_i^0(\boldsymbol{\pi}) - \pi_0 \iff \boldsymbol{\pi}^\top \boldsymbol{x} \leq \lambda_i^0(\boldsymbol{\pi}).$$

The last inequality above holds because $x_i = 0$ and, by definition, $\lambda_i^0(\boldsymbol{\pi}) = \max_{\boldsymbol{x}' \in X}\{\boldsymbol{\pi}^\top \boldsymbol{x}' : x_i' = 0\}$. As before, notice that this inequality is tight for $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}' \in X}\{\boldsymbol{\pi}^\top \boldsymbol{x}' : x_i' = 0\}$, i.e., $\boldsymbol{x}^* \in F(\boldsymbol{\pi}')$. Since $x_i' = 1$ for all $\boldsymbol{x}' \in F(\boldsymbol{\pi})$ (Lemma 1), the inequality is tight for all $\boldsymbol{x}' \in F(\boldsymbol{\pi})$, $\boldsymbol{x}^*$ is affinely independent to all points of $F(\boldsymbol{\pi})$, and therefore $\dim(F(\boldsymbol{\pi}')) \geq \dim(F(\boldsymbol{\pi})) + 1$.

Lastly, we demonstrate (3). We restrict to the case $i \in S^+(\boldsymbol{\pi})$; the other case is analogous. Given a fractional point $\bar{\boldsymbol{x}} \in [0,1]^n$ as defined above, we have $\boldsymbol{\pi}'^\top \bar{\boldsymbol{x}} = \boldsymbol{\pi}^\top \bar{\boldsymbol{x}} + \lambda_i(\boldsymbol{\pi})\bar{x}_i > \pi_0 + \lambda_i(\boldsymbol{\pi})\bar{x}_i \geq \pi_0 = \pi_0'$. ■

Example 2 Let $X = \{\boldsymbol{x} \in \{0,1\}^4 : 7x_1 + 5x_2 + 4x_3 + x_4 \leq 8\}$ and consider an inequality $x_1 + x_2 \leq 1$ supporting $\text{conv}(X)$. The d-slack is $\boldsymbol{\lambda}(\boldsymbol{\pi}) = (0,0,1,0)^\top$ and the lifted inequality with respect to $\lambda_3(\boldsymbol{\pi}) = 1$ is $\boldsymbol{\pi}'^\top \boldsymbol{x} = x_1 + x_2 + x_3 \leq 1$. Note that $\boldsymbol{\pi}'^\top \boldsymbol{x} \leq 1$ is facet-defining for $\text{conv}(X)$ and $\boldsymbol{\lambda}(\boldsymbol{\pi}') = \mathbf{0}$. □

4.2 Extracting Disjunctive Slacks from BDDs

Identifying d-slacks $\boldsymbol{\lambda}(\boldsymbol{\pi})$ is as a non-trivial task since we are required to solve $2n$ binary optimization problems, i.e., one for each component $i \in I$ and values 0 and 1. We now develop a procedure that generates the d-slacks by exploiting the network representation of a BDD $\mathcal{B} = (\mathcal{N}, \mathcal{A})$ for X. We also show that the procedure complexity is linear in the number of arcs $|\mathcal{A}|$ of $\mathcal{B}$.

First, for each arc $a \in \mathcal{A}$ with value $v_a \in \{0,1\}$ and source $s(a) \in \mathcal{N}_i$ for some $i \in I$, we associate a *length* of $\pi_i \cdot v_a$. Note that the longest $\mathbf{r} - \mathbf{t}$ path of $\mathcal{B}$ with respect to such lengths maximizes $\boldsymbol{\pi}^\top \boldsymbol{x}$ over X. Similarly, given the $\mathbf{r} - \mathbf{t}$ paths $\mathcal{P}$ of $\mathcal{B}$, let

$$\ell_a := \max\left\{\sum_{k=1}^{n} \pi_k \cdot v_{a_k} : p = (a_1, \ldots, a_n) \in \mathcal{P}, a_i = a\right\}$$

be the longest-path value conditioned on all paths that include a. Because each index $i \in I$ is uniquely associated with layer $\mathcal{N}_i$, it follows that

$$\lambda_i^j(\boldsymbol{\pi}) = \max_{a \in \mathcal{A}}\{\ell_a : s(a) \in \mathcal{N}_i, v_a = j\}, \quad \forall i \in I, \forall j \in \{0,1\},$$

and the final d-slacks are obtained by the differences $\lambda_i^0(\boldsymbol{\pi}) - \lambda_i^1(\boldsymbol{\pi})$ for all i.

The lengths ℓ_a are derived by performing two longest-path computations over $\mathcal{B}$. Specifically, let $\mathcal{A}^{\mathsf{in}}(u)$ and $\mathcal{A}^{\mathsf{out}}(u)$ be the set of incoming and outgoing arcs of a node $u \in \mathcal{N}$, respectively. The solution of the recursion

$$L^\downarrow(\boldsymbol{\pi}, \mathbf{r}) = 0,$$
$$L^\downarrow(\boldsymbol{\pi}, u) = \max_{a \in \mathcal{A}^{\mathsf{in}}(u)}\{L^\downarrow(\boldsymbol{\pi}, s(a)) + \pi_{i-1} \cdot v_a\}, \quad \forall u \in \mathcal{N}_i, \forall i \in \{2, \ldots, n+1\}$$

Algorithm 1 Sequential Combinatorial Lifting Procedure

1: **procedure** CombinatorialLifting($\langle \boldsymbol{\pi}, \pi_0 \rangle$, $\mathcal{B}$)
2: Calculate the disjunctive slacks $\boldsymbol{\lambda}(\boldsymbol{\pi})$
3: **while** $\boldsymbol{\lambda}(\boldsymbol{\pi}) \neq \mathbf{0}$ **do**
4: Choose $i \in I$ such that $\lambda_i(\boldsymbol{\pi}) \neq 0$
5: Apply Theorem 1 to calculate $\langle \boldsymbol{\pi}', \pi_0' \rangle$
6: Set $\langle \boldsymbol{\pi}, \pi_0 \rangle = \langle \boldsymbol{\pi}', \pi_0' \rangle$
7: Recalculate $\boldsymbol{\lambda}(\boldsymbol{\pi})$
8: **return** $\langle \boldsymbol{\pi}, \pi_0 \rangle$

provides the longest-path value $L^{\downarrow}(\boldsymbol{\pi}, u)$ from $\mathbf{r}$ to u, while the recursion

$$L^{\uparrow}(\boldsymbol{\pi}, u) = \max_{a \in \mathcal{A}^{\text{out}}(u)} \{L^{\uparrow}(\boldsymbol{\pi}, t(a)) + \pi_i \cdot v_a\}, \qquad \forall u \in \mathcal{N}_i,\ \forall i \in \{1, \dots, n\},$$
$$L^{\uparrow}(\boldsymbol{\pi}, \mathbf{t}) = 0$$

provides the longest-path value $L^{\uparrow}(\boldsymbol{\pi}, u)$ from u to $\mathbf{t}$. The values $L^{\downarrow}(\boldsymbol{\pi}, u)$ can be calculated via a top-down pass on $\mathcal{B}$, i.e., starting from $\mathbf{r}$ and considering one layer $\mathcal{N}_2, \dots, \mathcal{N}_{n+1}$ at a time. Analogously, the values $L^{\uparrow}(\boldsymbol{\pi}, u)$ are obtained via a bottom-up pass on $\mathcal{B}$, i.e., starting from $\mathbf{t}$ and considering one layer $\mathcal{N}_n, \mathcal{N}_{n-1}, \dots, \mathcal{N}_1$ at a time. For any arc $a = (s(a), t(a))$ such that $s(a) \in \mathcal{N}_i$,

$$\ell_a = L^{\downarrow}(\boldsymbol{\pi}, s(a)) + L^{\uparrow}(\boldsymbol{\pi}, t(a)) + \pi_i \cdot v_a.$$

Since each arc is traversed twice via the top-down and bottom-up passes, the complexity of the procedure is $\mathcal{O}(|A|)$.

4.3 Sequential Lifting and Dimension Implications

The lifting procedure detailed in Theorem 1 can be applied sequentially to strengthen an inequality. Specifically, we start with $\langle \boldsymbol{\pi}, \pi_0 \rangle$ satisfying our main assumptions (a)-(c). Next, we calculate the d-slacks, choose $i \in I$ such that $\lambda_i(\boldsymbol{\pi}) \neq 0$, and apply Theorem 1 to obtain the tuple $\langle \boldsymbol{\pi}', \pi_0' \rangle$ defining the lifted inequality. We re-apply this operation with the new $\langle \boldsymbol{\pi}', \pi_0' \rangle$, and repeat until $\lambda_i(\boldsymbol{\pi}) = 0$ for all $i \in I$. The procedure stops in a finite number of iterations since the face dimension increases after each rotation; see property (2) of Theorem 1. We summarize the procedure in Algorithm 1.

The choice of i in step 4 of Algorithm 1 is critical to the dimension of the resulting face, as illustrated in Example 3.

Example 3 Consider the set $X = \{x \in \{0,1\}^3 : 5x_1 + 2x_2 + 3x_3 \leq 6\}$ and inequality $\boldsymbol{\pi}^{\top}\boldsymbol{x} = x_1 + x_2 + x_3 \leq 2$ that supports $\text{conv}(X)$. We have $\boldsymbol{\lambda}(\boldsymbol{\pi}) = (1, -1, -1)^{\top}$ and the lifted inequality with respect to $\lambda_1(\boldsymbol{\pi}) = 1$ is $\boldsymbol{\pi'}^{\top}\boldsymbol{x} = 2x_1 + x_2 + x_3 \leq 2$ and has $\boldsymbol{\lambda}(\boldsymbol{\pi}') = \mathbf{0}$. The lifted inequality is not facet-defining since $\dim(\text{conv}(X)) = 3$ and $\dim(F(\boldsymbol{\pi}')) = 1$.

If we instead lift $x_1 + x_2 + x_3 \leq 2$ with respect to $\lambda_2(\boldsymbol{\pi}) = -1$ the lifted inequality is $\boldsymbol{\pi'}^{\top}\boldsymbol{x} = x_1 + x_3 \leq 1$ and $\boldsymbol{\lambda}(\boldsymbol{\pi}') = \mathbf{0}$. In this case, the lifted inequality is facet-defining since $\dim(F(\boldsymbol{\pi}')) = 2$. □

In order to understand the impact of the index choice, we first show in Lemma 2 a relationship between d-slacks and the dimension of the face. Specifically, the cardinality of $S^0(\boldsymbol{\pi})$ bounds $\dim(F(\boldsymbol{\pi}))$. We later use this result to gauge when the sequential procedure leads to a facet-defining inequality.

Lemma 2 *The dimension of a face $F(\boldsymbol{\pi})$ satisfies $\dim(F(\boldsymbol{\pi})) \leq |S^0(\boldsymbol{\pi})|$. Moreover, $|S^0(\boldsymbol{\pi})| = 0$ if $\dim(F(\boldsymbol{\pi})) = 0$.*

Proof For any $i \in S^-(\boldsymbol{\pi}) \cup S^+(\boldsymbol{\pi})$, the value of x_i is fixed at either 0 or 1 for all $\boldsymbol{x} \in F(\boldsymbol{\pi})$ according to Lemma 1. Thus, the dimension of $\dim(F(\boldsymbol{\pi}))$ is bounded by $|S^0(\boldsymbol{\pi})|$, since at most $|S^0(\boldsymbol{\pi})| + 1$ affinely independent points can be obtained from $F(\boldsymbol{\pi})$. Now, assume $S^0(\boldsymbol{\pi}) \neq \emptyset$ and $\dim(F(\boldsymbol{\pi})) > 0$. There exist $\boldsymbol{x}, \boldsymbol{x}' \in F(\boldsymbol{\pi})$ such that $x_i \neq x'_i$ for $i \in S^0(\boldsymbol{\pi})$. These two points are affinely independent and therefore $\dim(F(\boldsymbol{\pi})) \geq 1$. Thus, $S^0(\boldsymbol{\pi}) = \emptyset$ if $\dim(F(\boldsymbol{\pi})) = 0$. ■

Example 3 depicts a case where $|S^0(\boldsymbol{\pi})|$ increases faster than the number of affinely independent points in $F(\boldsymbol{\pi})$. In view of Lemma 2, we would like to choose i so that $|S^0(\boldsymbol{\pi})|$ increases at a slower rate, since each lifting operation increases $\dim(F(\boldsymbol{\pi}))$ by at least one according to Theorem 1-(2). We show in Theorem 2 that the slow increase of $|S^0(\boldsymbol{\pi})|$ occurs when there exists a unique slack with minimum non-zero absolute value.

Theorem 2 *Suppose there exists $i \notin S^0(\boldsymbol{\pi})$ such that $|\lambda_i(\boldsymbol{\pi})| < |\lambda_{i'}(\boldsymbol{\pi})|$ for all $i' \notin S^0(\boldsymbol{\pi})$ $(i' \neq i)$. Then, for $\langle \boldsymbol{\pi}', \pi'_0 \rangle$ obtained when lifting $\langle \boldsymbol{\pi}, \pi_0 \rangle$ with respect to $\lambda_i(\boldsymbol{\pi})$, $\dim(F(\boldsymbol{\pi}')) = \dim(F(\boldsymbol{\pi})) + 1$ and $|S^0(\boldsymbol{\pi}')| = |S^0(\boldsymbol{\pi})| + 1$.*

Proof From Lemma 1, it suffices to show that, for any $\boldsymbol{x} \in F(\boldsymbol{\pi}')$ and $i' \notin S^0(\boldsymbol{\pi})$ such that $i' \neq i$, we have: 1) $i' \in S^+(\boldsymbol{\pi})$ implies that $x_{i'} = 0$; and 2) $i' \in S^-(\boldsymbol{\pi})$ implies that $x_{i'} = 1$. In such cases, an index i' that was originally in $S^-(\boldsymbol{\pi})$ or $S^+(\boldsymbol{\pi})$ will remain in its original partition $S^-(\boldsymbol{\pi}')$ or $S^+(\boldsymbol{\pi}')$ for the lifted $\boldsymbol{\pi}'$. The statement then follows due to Theorem 1-(2) and Lemma 2.

We will focus our attention to the case $\lambda_i(\boldsymbol{\pi}) > 0$ (the others are analogous). For any $\boldsymbol{x} \in F(\boldsymbol{\pi}')$, we have by construction that $\boldsymbol{\pi}^\top \boldsymbol{x} = \pi'_0 - \lambda_i(\boldsymbol{\pi}) x_i \geq \pi_0 - \lambda_i(\boldsymbol{\pi})$. Assume, for the purpose of a contradiction, that $x_{i'} = 1$ and that $\lambda_{i'}(\boldsymbol{\pi}) > 0$. Thus, $\lambda^1_{i'}(\boldsymbol{\pi}) \geq \boldsymbol{\pi}^\top \boldsymbol{x} \geq \pi_0 - \lambda_i(\boldsymbol{\pi})$. Moreover, $\lambda^0_{i'}(\boldsymbol{\pi}) = \pi_0$. This implies that $\lambda_{i'}(\boldsymbol{\pi}) = \lambda^0_{i'}(\boldsymbol{\pi}) - \lambda^1_{i'}(\boldsymbol{\pi}) \leq \pi_0 - \pi_0 + \lambda_i(\boldsymbol{\pi}) \leq \lambda_i(\boldsymbol{\pi})$ and hence $0 < \lambda_{i'}(\boldsymbol{\pi}) \leq \lambda_i(\boldsymbol{\pi})$. This cannot hold since $|\lambda_i(\boldsymbol{\pi})| < |\lambda_{i'}(\boldsymbol{\pi})|$.

Similarly, assume that $\lambda_{i'}(\boldsymbol{\pi}) < 0$ and $x_i = 0$. Then, $\lambda^0_{i'}(\pi) \geq \pi_0 - \lambda_i(\boldsymbol{\pi})$ and $\lambda^1_{i'}(\boldsymbol{\pi}) = \pi_0$. This implies that $\lambda_{i'}(\boldsymbol{\pi}) = \lambda^0_{i'}(\boldsymbol{\pi}) - \lambda^1_{i'}(\boldsymbol{\pi}) \geq \pi_0 - \lambda_i(\boldsymbol{\pi}) - \pi_0 = -\lambda_i(\boldsymbol{\pi})$. Thus, $0 > \lambda_{i'}(\boldsymbol{\pi}) \geq -\lambda_i(\boldsymbol{\pi})$. This contradicts $|\lambda_i(\boldsymbol{\pi})| < |\lambda_{i'}(\boldsymbol{\pi})|$. ■

Theorem 2 provides a simple choice rule based on picking i with the minimum absolute slack. It also indicates when this rule will converge to a facet-defining inequality. We formalize it in Corollary 1 below, which can be derived as a direct consequence of Theorem 2.

Corollary 1 *If* $\dim(F(\boldsymbol{\pi})) = |S^0(\boldsymbol{\pi})|$*, the sequential lifting procedure (Algorithm 1) with the minimum slack absolute rule produces a facet-defining inequality if, at each lifting iteration except the last, the chosen* $i \in I$ *is such that* $|\lambda_i(\boldsymbol{\pi})| < |\lambda_{i'}(\boldsymbol{\pi})|$ *for all* $i' \notin S^0(\boldsymbol{\pi})$ $(i' \neq i)$*.*

Finally, we note that, in general, it may not be possible to achieve a facet-defining inequality. For example, all non-zero d-slacks can have the same absolute value and the cardinality of $|S^0(\boldsymbol{\pi})|$ might increase by more than one while the dimension of $F(\boldsymbol{\pi})$ does not (see Example 3).

4.4 Relationship with Lifting based on Disjunctive Programming

We now formalize the connection between our lifting methodology and the n-step lifting procedure by Perregaard and Balas [48] mentioned in §3. Assume that $X = \{A\boldsymbol{x} \leq \boldsymbol{b},\ \boldsymbol{x} \in \{0,1\}^n\}$ for a matrix A and vector $\boldsymbol{b}$ of appropriate dimensions. We consider a relaxation of the form DP-L that includes one disjunctive term for each index $i \in I$ and removes all integrality constraints:

$$\max_{\boldsymbol{x}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} :\ A\boldsymbol{x} \leq \boldsymbol{b},\ \bigvee_{i' \in I} (x_{i'} \leq 0) \vee (x_{i'} \geq 1),\ \boldsymbol{x} \in [0,1]^n \right\}. \tag{DP-L}$$

Proposition 1 below shows that, for DP-L, the optimal rotation parameter in the n-step lifting is such that $\gamma^* = |\lambda_i(\boldsymbol{\pi})|$ when using the individual binary disjunctions as target inequalities.

Proposition 1 *Suppose* $\lambda_i(\boldsymbol{\pi}) \neq 0$ *for some* $i \in I$*. Then,* $\gamma^* = |\lambda_i(\boldsymbol{\pi})|$ *if we employ either* $x_i \leq 0$ *or* $x_i \geq 1$ *as a target inequality in the* n*-step procedure.*

Proof Consider the case when $i \in S^+(\boldsymbol{\pi})$ and let $\boldsymbol{e}_i$ be the i-th column of an $n \times n$ identity matrix. Since all $\boldsymbol{x} \in F(\boldsymbol{\pi})$ have $x_i = 0$, $\widetilde{\boldsymbol{\pi}}^\top \boldsymbol{x} = \boldsymbol{e}_i^\top \boldsymbol{x} = x_i \leq 0$ is an invalid target inequality for $\mathrm{conv}(X)$ and satisfies $\widetilde{\boldsymbol{\pi}}^\top \boldsymbol{x} = x_i = 0$ for all $\boldsymbol{x} \in F(\boldsymbol{\pi})$. The system that defines γ^* is therefore

$$\begin{aligned} \gamma^* = \min_{\boldsymbol{x} \in [0,1]^n, x_0 \geq 0} \{\ & \pi_0 x_0 - \boldsymbol{\pi}^\top \boldsymbol{x} :\ A\boldsymbol{x} \leq \boldsymbol{b},\ -x_i = -1, \\ & \bigvee_{i' \in I} (x_{i'} \leq 0) \vee (-x_{i'} + x_0 \leq 0)\}. \end{aligned} \tag{1a}$$

It follows from (1a) and $x_i = 1$ that $x_0 \leq 1$. Without loss of generality, we consider $x_0 = 1$. The system reduces to:

$$\begin{aligned} \gamma^* &= \min\left\{\pi_0 x_0 - \boldsymbol{\pi}^\top \boldsymbol{x} : A\boldsymbol{x} \leq \boldsymbol{b},\ x_i = 1,\ \boldsymbol{x} \in \{0,1\}^n,\ x_0 = 1\right\} \\ &= \pi_0 - \max\{\boldsymbol{\pi}^\top \boldsymbol{x} : \boldsymbol{x} \in X,\ x_i = 1\} \\ &= \lambda_i^0(\boldsymbol{\pi}) - \lambda_i^1(\boldsymbol{\pi}) = \lambda_i(\boldsymbol{\pi}). \end{aligned}$$

The last equality comes from $i \in S^+(\boldsymbol{\pi})$ and $\lambda_i^0(\boldsymbol{\pi}) = \pi_0$. The proof for $i \in S^-(\boldsymbol{\pi})$ or with target inequality $\widetilde{\boldsymbol{\pi}}^\top \boldsymbol{x} = x_i \geq 1$ is analogous. ■

Proposition 1 indicates when techniques are equivalent. By taking $\widetilde{\boldsymbol{\pi}}^\top \boldsymbol{x} = x_i \leq 0$ as the target inequality, we obtain $\gamma^* = \lambda_i(\boldsymbol{\pi}) > 0$. The rotated inequality $(\boldsymbol{\pi} + \gamma^* \widetilde{\boldsymbol{\pi}})^\top \boldsymbol{x} = (\boldsymbol{\pi} + \lambda_i(\boldsymbol{\pi})\boldsymbol{e}_i)^\top \boldsymbol{x} \leq \pi_0$ is equivalent to the lifted inequality in Theorem 1. Similarly, using target inequality $\widetilde{\boldsymbol{\pi}}^\top \boldsymbol{x} = -x_i \leq -1$ would result in $\gamma^* = -\lambda_i(\boldsymbol{\pi}) > 0$. Then, the rotated and lifted inequalities are equivalent, i.e, $\boldsymbol{\pi} + \gamma^* \widetilde{\boldsymbol{\pi}} = \boldsymbol{\pi} + \lambda_i(\boldsymbol{\pi})\boldsymbol{e}_i$ and $\pi_0 + \gamma^* \widetilde{\pi}_0 = \pi_0 + \lambda_i(\boldsymbol{\pi})$.

While the techniques are equivalent in this restricted case, our approach is valid for any binary set X and, thus, can handle models where a BDD (or BDD relaxation) is a more advantageous representation in comparison to a linear description of X [15]. We also note that the BDD network structure allows us to efficiently compute the disjunctive terms in a combinatorial fashion.

5 Combinatorial Cutting-Plane Algorithm

While the BDD-based lifting procedure developed in §4 can enhance inequalities from any cutting-plane methodology, we now exploit similar concepts to derive new valid inequalities for X based on the network structure of $\mathcal{B}$. In particular, we design inequalities that separate points from X by only relying on the combinatorial structure encoded by $\mathcal{B}$. Thus, no other specific structure (e.g., linearity, submodularity, or gradient information) is required.

We assume, as before, that we are given an exact BDD $\mathcal{B}$ for X. Our cutting-plane method is based on an alternative linear description of $\mathcal{B}$ as an extended capacitated flow problem. We present this formulation in §5.1 and our BDD-based cut generation linear program in §5.2. For cases where solving such model is not computationally practical, in §5.3 we develop a weaker but more efficient combinatorial cutting-plane method based on a max-flow/min-cut problem over $\mathcal{B}$. Finally, we show in §5.4 the relationship between our approach and existing BDD cutting-plane techniques [28,51].

5.1 BDD Polytope

Existing BDD-based cut generation procedures [28,40,51] rely on the network-flow formulation $\mathrm{NF}(\mathcal{B})$ introduced by Behle [13], described as follows:

$$\begin{aligned}
\mathrm{NF}(\mathcal{B}) := \{(\boldsymbol{x}; \boldsymbol{y}) \in [0,1]^n \times \mathbb{R}_+^{|\mathcal{A}|} : \\
\sum_{a \in \mathcal{A}^{\mathsf{out}}(u)} y_a - \sum_{a \in \mathcal{A}^{\mathsf{in}}(u)} y_a = 0, \qquad \forall u \in \mathcal{N} \setminus \{\mathbf{r}, \mathbf{t}\}, \qquad (2a) \\
\sum_{a \in \mathcal{A}^{\mathsf{out}}(\mathbf{r})} y_a = \sum_{a \in \mathcal{A}^{\mathsf{in}}(\mathbf{t})} y_a = 1, \qquad (2b) \\
\sum_{a \in \mathcal{A} : s(a) \in \mathcal{N}_i, v_a = 1} y_a = x_i, \qquad \forall i \in I \Big\}. \qquad (2c)
\end{aligned}$$

Equalities (2a) and (2b) are balance-of-flow constraints over $\mathcal{B}$. Constraint (2c) links the arcs of $\mathcal{B}$ with solutions $\boldsymbol{x}$. In particular, the polytope $\mathrm{NF}(\mathcal{B})$

projected over the $\boldsymbol{x}$ variables is equivalent to the convex hull of all solutions represented by $\mathcal{B}$, i.e., $\text{Proj}_{\text{x}}(\text{NF}(\mathcal{B})) = \text{conv}(X)$.

We propose an alternative formulation to $\text{NF}(\mathcal{B})$ and use it to define our cutting-plane algorithm. The new formulation, here denoted by $\text{JNF}(\mathcal{B})$, corresponds to a joint capacitated network-flow polytope. The new model maintains the flow conservation constraints, (2a) and (2b), and replaces (2c) with (3a) and (3b) below. Both inequalities enforce a common capacity for arcs in a layer with the same value. Proposition 2 shows that the two formulations are equivalent and, thus, $\text{Proj}_{\text{x}}(\text{JNF}(\mathcal{B})) = \text{conv}(X)$.

$$\begin{aligned}\text{JNF}(\mathcal{B}) := \{(\boldsymbol{x};\boldsymbol{y}) \in [0,1]^n \times \mathbb{R}_+^{|\mathcal{A}|} : (2a)-(2b),& \\ \sum_{a\in\mathcal{A}: s(a)\in\mathcal{N}_i, v_a=1} y_a \leq x_i, \qquad & \forall i \in I, \qquad (3a)\\ \sum_{a\in\mathcal{A}: s(a)\in\mathcal{N}_i, v_a=0} y_a \leq 1 - x_i, \qquad & \forall i \in I\ \}. \qquad (3b)\end{aligned}$$

Proposition 2 *$JNF(\mathcal{B}) = NF(\mathcal{B})$.*

Proof Consider $(\boldsymbol{x}';\boldsymbol{y}') \in \text{NF}(\mathcal{B})$, so $(\boldsymbol{x}';\boldsymbol{y}')$ satisfies (2a) and (2b). Since (2c) holds, $(\boldsymbol{x}';\boldsymbol{y}')$ also satisfies (3a). From the flow conservation constraints, (2a) and (2b), the flow traversing each layer $i \in I$ in $\mathcal{B}$ is exactly one, i.e.,

$$\begin{aligned}\sum_{a\in\mathcal{A}:\ s(a)\in\mathcal{N}_i} y'_a = 1 \quad &\Rightarrow \quad \sum_{a\in\mathcal{A}: s(a)\in\mathcal{N}_i:\ v_a=1} y'_a + \sum_{a\in\mathcal{A}: s(a)\in\mathcal{N}_i:\ v_a=0} y'_a = 1\\ &\Rightarrow \quad \sum_{a\in\mathcal{A}: s(a)\in\mathcal{N}_i:\ v_a=0} y'_a = 1 - x'_i.\end{aligned}$$

Then, $(\boldsymbol{x}';\boldsymbol{y}')$ satisfies (3b) and $(\boldsymbol{x}';\boldsymbol{y}') \in \text{JNF}(\mathcal{B})$. Consider now $(\boldsymbol{x}';\boldsymbol{y}') \in \text{JNF}(\mathcal{B})$. Since flows traversing a layer sum to one, constraints (3a) and (3b) are satisfied as equalities and therefore (2c) holds for $(\boldsymbol{x}';\boldsymbol{y}')$. ■

5.2 General BDD Flow Cuts

Our cutting-plane procedure formulates a max-flow optimization problem over $\text{JNF}(\mathcal{B})$ to identify and separate points $\boldsymbol{x}' \notin \text{conv}(X)$, given by (4) below:

$$z(\mathcal{B};\boldsymbol{x}') := \max_{\boldsymbol{y}\in\mathbb{R}_+^{|\mathcal{A}|}} \left\{ \sum_{a\in\mathcal{A}^{\text{out}}(\mathbf{r})} y_a :\ (2a), (3a)-(3b), \boldsymbol{x} = \boldsymbol{x}' \right\}. \qquad (4)$$

We note that (4) omits the constraint enforcing the flow to be equal to one as in (2b). We argue in Lemma 3 that the condition $z(\mathcal{B};\boldsymbol{x}') = 1$ is necessary and sufficient to check if $\boldsymbol{x}'$ belongs to $\text{conv}(X)$.

Lemma 3 *$\boldsymbol{x}' \in \text{conv}(X)$ if and only if $z(\mathcal{B};\boldsymbol{x}') = 1$.*

Proof Constraints (3a) and (3b) enforce that the flow in each layer i is at most $x'_i + 1 - x'_i = 1$. Thus, $z(\mathcal{B}; \boldsymbol{x}') \leq 1$. Consider $\boldsymbol{x}' \in \text{conv}(X)$. From Proposition 2, there exists $\boldsymbol{y}' \in \mathbb{R}_+^{|\mathcal{A}|}$ such that $\sum_{a \in \mathcal{A}^{\text{out}}(\mathbf{r})} y'_a = 1$ and therefore $z(\mathcal{B}; \boldsymbol{x}') = 1$. For the converse, suppose $z(\mathcal{B}; \boldsymbol{x}') = 1$. It follows that there exists $\boldsymbol{y}' \in \mathbb{R}_+^{|\mathcal{A}|}$ such that $(\boldsymbol{x}'; \boldsymbol{y}') \in \text{JNF}(\mathcal{B}) = \text{NF}(\mathcal{B})$, so $\boldsymbol{x}' \in \text{conv}(X)$. ■

Our BDD-based CGLP uses the dual of (4) to generate valid inequalities that cut off $\boldsymbol{x}' \notin \text{conv}(X)$. Consider $\boldsymbol{\omega} \in \mathbb{R}^{|\mathcal{N}|}$ as the dual variables associated with constraints (2a), and $\boldsymbol{\nu}, \boldsymbol{\eta} \in \mathbb{R}_+^n$ as the dual variables for (3a) and (3b), respectively. The resulting model is

$$\min_{\boldsymbol{\omega},\boldsymbol{\nu},\boldsymbol{\eta}} \quad \sum_{i \in I} x'_i \nu_i + \sum_{i \in I} (1 - x'_i)\eta_i \tag{BDD-CGLP}$$

$$\text{s.t.} \quad \omega_{t(a)} - \omega_{s(a)} + v_a \nu_i + (1 - v_a)\eta_i \geq 0, \quad \forall i \in I, a \in \mathcal{A}, s(a) \in \mathcal{N}_i, \tag{5a}$$

$$\omega_{t(a)} + v_a \nu_1 + (1 - v_a)\eta_1 \geq 1, \quad \forall a \in \mathcal{A}^{\mathsf{out}}(\mathbf{r}), \tag{5b}$$

$$-\omega_{s(a)} + v_a \nu_n + (1 - v_a)\eta_n \geq 0, \quad \forall a \in \mathcal{A}^{\mathsf{in}}(\mathbf{t}), \tag{5c}$$

$$\boldsymbol{\omega} \in \mathbb{R}^{|\mathcal{N}|}, \ \boldsymbol{\nu}, \boldsymbol{\eta} \in \mathbb{R}_+^n. \tag{5d}$$

Let $w(\mathcal{B}; \boldsymbol{x}')$ be the optimal solution value of BDD-CGLP. Strong duality and Lemma 3 imply that we can identify if a point $\boldsymbol{x}'$ belongs to $\text{conv}(X)$ if $w(\mathcal{B}; \boldsymbol{x}') = 1$. Furthermore, we can use the optimal solution $(\boldsymbol{\nu}^*; \boldsymbol{\eta}^*)$ to create a valid cut when $w(\mathcal{B}; \boldsymbol{x}') < 1$. Specifically, the cut is given by

$$\sum_{i \in I} x_i \nu_i^* + \sum_{i \in I} (1 - x_i)\eta_i^* \geq 1. \tag{6}$$

Theorem 3 shows that the set of all cuts of the form (6) describes $\text{conv}(X)$.

Theorem 3 *Let $\Lambda(\mathcal{B})$ be the set of extreme points of the BDD-CGLP polyhedron defined by (5a)-(5d). Furthermore, let $P_{\mathcal{B}}$ be the set of points $\boldsymbol{x} \in [0,1]^n$ that satisfy (6) for all $(\boldsymbol{\nu}; \boldsymbol{\eta}) \in \text{Proj}_{\boldsymbol{\nu},\boldsymbol{\eta}}(\Lambda(\mathcal{B}))$. Then, $\text{conv}(X) = P_{\mathcal{B}}$.*

Proof Consider a point $\boldsymbol{x}' \in \text{conv}(X)$. Lemma 3 guarantees that $z(\mathcal{B}; \boldsymbol{x}') = w(\mathcal{B}; \boldsymbol{x}') = 1$, so constraint (6) holds for any extreme point of BDD-CGLP. Now consider a point $\boldsymbol{x}' \in P_{\mathcal{B}}$. Since $\boldsymbol{x}'$ satisfies (6) for all extreme points in $\Lambda(\mathcal{B})$, we have that $w(\mathcal{B}; \boldsymbol{x}') \geq 1$ and, thus, $w(\mathcal{B}; \boldsymbol{x}') = z(\mathcal{B}; \boldsymbol{x}') = 1$. Finally, using Lemma 3, we have that $\boldsymbol{x}' \in \text{conv}(X)$. ■

Thus, our cutting-plane procedure separates points $\boldsymbol{x}' \notin \text{conv}(X)$ by solving BDD-CGLP. The procedure returns a cut of the form (6) where $(\boldsymbol{\nu}^*; \boldsymbol{\eta}^*)$ is an optimal solution of $w(\mathcal{B}; \boldsymbol{x}')$.

5.3 Combinatorial BDD Flow Cuts

The above cutting-plane procedure requires solving a linear program with $|\mathcal{A}|$ constraints and $|\mathcal{N}| + 2n$ variables. Obtaining $w(\mathcal{B}; \boldsymbol{x}')$, thus, could be

computationally expensive for instances where $\mathcal{B}$ is large. We propose now an alternative cut-generation procedure based on BDD-CGLP that involves a combinatorial and more efficient max-flow solution over $\mathcal{B}$:

First, we consider a reformulation of JNF($\mathcal{B}$) where the joint capacity constraints are replaced by individual constraints for each arc, i.e., a standard capacitated network flow polytope over $\mathcal{B}$:

$$\begin{aligned}\text{CNF}(\mathcal{B}) := \{(\boldsymbol{x};\boldsymbol{y}) \in [0,1]^n \times \mathbb{R}_+^{|\mathcal{A}|} : (2a) - (2b),\\ y_a \le x_i, \qquad \forall a \in \mathcal{A}, s(a) \in \mathcal{N}_i, v_a = 1, i \in I, \qquad (7a)\\ y_a \le 1 - x_i, \qquad \forall a \in \mathcal{A}, s(a) \in \mathcal{N}_i, v_a = 0, i \in I\}. \qquad (7b)\end{aligned}$$

Proposition 3 *JNF($\mathcal{B}$) $\subseteq$ CNF($\mathcal{B}$). Moreover, for any integer $\boldsymbol{x} \notin \text{conv}(X)$, we have that $\boldsymbol{x} \notin \text{Proj}_x(\text{CNF}(\mathcal{B}))$.*

Proof Consider $\boldsymbol{x}' \in \text{JNF}(\mathcal{B})$. By construction, $\boldsymbol{x}'$ satisfies (2a)-(2b). Since $\boldsymbol{x}'$ satisfies (3a) and (3b), it follows that $\boldsymbol{x}'$ holds for (7a) and (7b).

Now take an integer point $\boldsymbol{x}' \notin \text{conv}(X)$ and $\boldsymbol{y}' \in \mathbb{R}_+^{|\mathcal{A}|}$ such that $(\boldsymbol{x}';\boldsymbol{y}')$ satisfies (2a), (7a)-(7b). Notice that such a $\boldsymbol{y}'$ exists (e.g., $\boldsymbol{y}' = \boldsymbol{0}$). By construction, there is no path $p \in \mathcal{P}$ associated with $\boldsymbol{x}'$. From constraints (7a)-(7b), in any path $p \in \mathcal{P}$ there exists an arc $a \in p$ with capacity zero (i.e., $y_a \le 0$). We can then deduce that $\boldsymbol{y}' = \boldsymbol{0}$, therefore $(\boldsymbol{x}';\boldsymbol{y}')$ violates (2b). Finally, for any $\boldsymbol{x}' \in \{0,1\}^n \setminus \text{conv}(X)$ there is no $\boldsymbol{y}' \in \mathbb{R}_+^{|\mathcal{A}|}$ such that $(\boldsymbol{x}';\boldsymbol{y}') \in \text{CNF}(\mathcal{B})$. ■

Proposition 3 shows that for any integer point $\boldsymbol{x}'$, $\boldsymbol{x}' \notin X$ implies $\boldsymbol{x}' \notin \text{Proj}_x(\text{CNF}(\mathcal{B}))$. Example 4 illustrates that, conversely, there might exist fractional points $\boldsymbol{x}' \notin \text{conv}(X)$ such that $\boldsymbol{x}' \in \text{Proj}_x(\text{CNF}(\mathcal{B}))$, and hence CNF($\mathcal{B}$) is a weaker representation.

Example 4 Consider our example $X = \{\boldsymbol{x} \in \{0,1\}^4 : 7x_1+5x_2+4x_3+x_4 \le 8\}$, a fractional point $\boldsymbol{x}' = (0.4, 0.6, 0.4, 1)$, and the exact BDD $\mathcal{B}_1$ in Figure 1. It is easy to see that $\boldsymbol{x}' \notin \text{conv}(X)$ since $7x'_1+5x'_2+4x'_3+x'_4 = 8.4 \ge 8$. However, there exists a $\boldsymbol{y}' \in \mathbb{R}_+^{|\mathcal{A}|}$ such that $(\boldsymbol{x}', \boldsymbol{y}') \in \text{CNF}(\mathcal{B})$ with value $y_{(\mathbf{r},u_1)} = 0.6$, $y_{(\mathbf{r},u_2)} = 0.4$, $y_{(u_1,u_4)} = 0.2$, $y_{(u_1,u_3)} = 0.4$, $y_{(u_2,u_4)} = 0.4$, $y_{(u_3,u_4)} = 0.4$, $y_{(u_4,u_5)} = 0.6$, $y_{(u_5,\mathbf{t})} = 1$, and all other arcs with flow equal to zero. □

Similar to the general BDD flow cuts, we use the dual of the max-flow version of CNF($\mathcal{B}$) to identify points that do not belong to conv(X). Consider $\boldsymbol{\omega} \in \mathbb{R}^{|\mathcal{N}|}$ as the dual variables associated with constraints (2a) and $\boldsymbol{\alpha}$ the dual variables associated with constraints (7a)-(7b). Then, the separation problem for the alternative BDD cuts is as follow:

$$\begin{aligned}\min_{\boldsymbol{\omega},\boldsymbol{\alpha}} \quad & \sum_{i \in I} \left(\sum_{a \in \mathcal{A}: s(a) \in \mathcal{N}_i, v_a = 1} x'_i \alpha_a \;+ \sum_{a \in \mathcal{A}: s(a) \in \mathcal{N}_i, v_a = 0} (1 - x'_i)\alpha_a \right) \qquad \text{(CN-CGLP)}\\ \text{s.t.} \quad & \omega_{t(a)} - \omega_{s(a)} + \alpha_a \ge 0, && \forall i \in I, a \in \mathcal{A}, s(a) \in \mathcal{N}_i,\\ & \omega_{t(a)} + \alpha_a \ge 1, && \forall a \in \mathcal{A}^{\text{out}}(\mathbf{r}),\\ & -\omega_{s(a)} + \alpha_a \ge 0, && \forall a \in \mathcal{A}^{\text{in}}(\mathbf{t}),\\ & \boldsymbol{\omega} \in \mathbb{R}^{|\mathcal{N}|},\ \boldsymbol{\alpha} \in \mathbb{R}_+^{|\mathcal{A}|}.\end{aligned}$$

Let $w^{\mathrm{r}}(\mathcal{B};\boldsymbol{x}')$ be the optimal solution value of CN-CGLP. Proposition 3 implies that for any $\boldsymbol{x}' \in \mathrm{conv}(X_{\mathcal{B}})$, $w^{\mathrm{r}}(\mathcal{B};\boldsymbol{x}') = 1$. It follows that inequality (9) holds for any $\boldsymbol{x} \in \mathrm{conv}(X_{\mathcal{B}})$, where $\boldsymbol{\alpha}^*$ is optimal to CN-CGLP:

$$\sum_{i\in I}\left(\sum_{a\in\mathcal{A}:s(a)\in\mathcal{N}_i,v_a=1} x_i\alpha^*_a + \sum_{a\in\mathcal{A}:s(a)\in\mathcal{N}_i,v_a=0}(1-x_i)\alpha^*_a\right) \geq 1. \tag{9}$$

Of important note is that CN-CGLP is a classical min-cut problem, i.e., we are searching for a maximum-capacity arc cut in the network that certifies that a point does not belong to the convex hull of X. While the resulting inequalities are not as strong as the general BDD cuts from BDD-CGLP, we can leverage max-flow/min-cut combinatorial algorithms to solve it more efficiently in the size of the BDD. Several algorithms are readily available to that end [3] and provide both primal and dual solutions to CN-CGLP.

Furthermore, another consequence of the design of such cuts is that their strength depends on the BDD size. That is, two BDDs $\mathcal{B}$ and $\mathcal{B}'$ encoding the same set might generate different combinatorial flow cuts because of distinct min-cut solutions. We show in Theorem 4 that the reduced BDD, which is unique, generates the tightest CNF($\mathcal{B}$) formulation and is hence critical in such a formulation. We note that a reduced BDD can be generated in polynomial time in $\mathcal{B}'$ for any $\mathcal{B}'$ representing the desired solution set [22].

Theorem 4 *Let $\mathcal{B}^{\mathrm{r}} = (\mathcal{N}^{\mathrm{r}}, \mathcal{A}^{\mathrm{r}})$ be the reduced version of $\mathcal{B}$, i.e., $X_{\mathcal{B}^{\mathrm{r}}} = X_{\mathcal{B}}$, and for each layer $i \in I$, $|\mathcal{N}^{\mathrm{r}}_i| \leq |\mathcal{N}_i|$. Then, CNF($\mathcal{B}^{\mathrm{r}}$) $\subseteq$ CNF($\mathcal{B}$).*

Proof Consider $\mathcal{P}^{\mathrm{r}}$ to be the set of $\mathbf{r}-\mathbf{t}$ paths in $\mathcal{B}^{\mathrm{r}}$. First, $X_{\mathcal{B}^{\mathrm{r}}} = X_{\mathcal{B}}$ implies that, for any $\mathbf{r}-\mathbf{t}$ path $p \in \mathcal{P}$, there exists a unique $\mathbf{r}-\mathbf{t}$ path $p' \in \mathcal{P}^{\mathrm{r}}$ such that $\boldsymbol{x}^p = \boldsymbol{x}^{p'}$. Thus, we will consider that the set of paths in both BDDs are equivalent, i.e., $\mathcal{P}^{\mathrm{r}} = \mathcal{P}$.

Let $\mathcal{A}_i = \{a \in \mathcal{A}\colon s(a) \in \mathcal{N}_i\}$ and $\mathcal{A}^{\mathrm{r}}_i = \{a \in \mathcal{A}^{\mathrm{r}}\colon s(a) \in \mathcal{N}^{\mathrm{r}}_i\}$. Since $\mathcal{B}^{\mathrm{r}}$ is unique, there exists a unique surjective function $f_i : \mathcal{A}_i \to \mathcal{A}^{\mathrm{r}}_i$ that maps arcs from $\mathcal{B}$ to $\mathcal{B}^{\mathrm{r}}$ for each layer $i \in I$. Thus, for every arc $a \in \mathcal{A}^{\mathrm{r}}_i$, let us define the pre-image of f_i as $f_i^{-1}(a) := \{a' \in \mathcal{A}_i :\ f(a') = a\}$, i.e., the subset of arcs in $\mathcal{A}_i$ that map to arc $a \in \mathcal{A}^{\mathrm{r}}_i$. Next, denote by $\Gamma(\mathcal{B};a) := \{p \in \mathcal{B} : a \in p\}$ the set of paths in a BDD that traverse an arc $a \in \mathcal{A}$. From the construction procedure of $\mathcal{B}^{\mathrm{r}}$ given $\mathcal{B}$ [22], $\Gamma(\mathcal{B}^{\mathrm{r}};a) = \bigcup_{a'\in f_i^{-1}(a)} \Gamma(\mathcal{B};a')$ for all $a \in \mathcal{A}^{\mathrm{r}}_i$, i.e., the set of $\mathbf{r}-\mathbf{t}$ paths passing through a is equivalent to the set of $\mathbf{r}-\mathbf{t}$ paths passing through all the arcs in $f_i^{-1}(a)$.

Now consider the path formulation of CNF($\mathcal{B}$), CNF$^P(\mathcal{B})$. It suffices to show that CNF$^P(\mathcal{B}^{\mathrm{r}}) \subseteq$ CNF$^P(\mathcal{B})$. Since the paths in $\mathcal{B}$ and $\mathcal{B}^{\mathrm{r}}$ are equivalent, we will consider equivalent variables $\boldsymbol{w}$ for CNF$^P(\mathcal{B}^{\mathrm{r}})$ and CNF$^P(\mathcal{B})$.

$$\begin{aligned}
\mathrm{CNF}^P(\mathcal{B}) := \{(\boldsymbol{x};\boldsymbol{w}) \in [0,1]^n \times \mathbb{R}_+^{|X(\mathcal{B})|} :\ & \\
\sum_{p\in\mathcal{P}:a\in p} w_p \leq x_i, \qquad & \forall a \in \mathcal{A}_i, v_a = 1, i \in I, && \text{(10a)}\\
\sum_{p\in\mathcal{P}:a\in p} w_p \leq 1 - x_i, \qquad & \forall a \in \mathcal{A}_i, v_a = 0, i \in I\}. && \text{(10b)}
\end{aligned}$$

Using the path equivalence $\Gamma(\mathcal{B}^{\mathsf{r}};a) = \bigcup_{a' \in f_i^{-1}(a)} \Gamma(\mathcal{B};a')$ for any $a \in \mathcal{A}_i^{\mathsf{r}}$ and $i \in I$, we have that

$$\sum_{p \in \mathcal{P}^{\mathsf{r}}: a \in p} w_p = \sum_{a' \in f_i^{-1}(a)} \sum_{p \in \mathcal{P}: a' \in p} w_p, \qquad \forall a \in \mathcal{A}_i^{\mathsf{r}},\ i \in I.$$

Thus, constraints (10a) and (10b) of $\text{CNF}^P(\mathcal{B}^{\mathsf{r}})$ are tighter since they restrict more paths than for the case of $\text{CNF}^P(\mathcal{B})$. This implies that, for any $(\boldsymbol{x}';\boldsymbol{w}') \in \text{CNF}^P(\mathcal{B}^{\mathsf{r}})$, $(\boldsymbol{x}';\boldsymbol{w}') \in \text{CNF}^P(\mathcal{B})$. ■

Theorem 4 indicates that the reduced BDD can separate more points than any other BDD representing the same solution set. We also note in passing that the variable ordering plays a role on the size of the BDD and, hence, on the effectiveness of the weaker combinatorial BDD flow cuts. Investigating variable orderings for specific problem classes and how they impact the cuts (theoretically and computationally) may lead to new research avenues.

5.4 Relationship with Existing BDD Cut Generation Procedures

The two existing BDD-based CGLPs rely on dual reformulations of $\text{NF}(\mathcal{B})$, and, thus, also describe $\text{conv}(X)$ [51,28]. These techniques rely on additional information: Tjandraatmadja et al. (2019) [51] CGLP requires an interior point of X and Davarnia et al. (2020) [28] must incorporate possibly non-linear normalization constraints. In contrast, BDD-CGLP exploits the structure of $\mathcal{B}$ directly to describe $\text{conv}(X)$. We now detail these two BDD-based CGLPs and highlight the main theoretical differences to BDD-CGLP.

Consider $\boldsymbol{\omega} \in \mathbb{R}^{|\mathcal{N}|}$ as the dual variables associated with constraints (2a) and (2b), and $\boldsymbol{\theta} \in \mathbb{R}^n$ as the dual variables associated with (2c). The two BDD-based CGLP models employ flow inequalities of the form

$$\omega_{t(a)} - \omega_{s(a)} + \theta_i v_a \geq 0, \qquad \forall i \in I, a \in \mathcal{A}, s(a) \in \mathcal{N}_i. \tag{11}$$

Notice that (11) resembles the flow inequalities (5a)-(5c) of BDD-CGLP. However, our flow constraints use two sets of positive dual variables for each BDD layer (i.e., $\boldsymbol{\nu}, \boldsymbol{\eta} \in \mathbb{R}_+^n$) instead of the single unbounded set of variables $\boldsymbol{\theta} \in \mathbb{R}^n$. This difference emerges because (2c) only bounds the arc flow variables $\boldsymbol{y} \in \mathbb{R}_+^{|\mathcal{A}|}$ with value $v_a = 1$, while our joint-capacity constraints (3a)-(3b) bound all variables $\boldsymbol{y}$. This is one reason, e.g., why BDD-CGLP does not require any normalization as in previous techniques.

Formally, Tjandraatmadja et al. (2019) [51] propose a BDD-based CGLP (12) to generate target cuts. Their CGLP derives a valid inequality that intersects the ray passing through an interior point $\boldsymbol{u} \in \text{conv}(X)$ and the fractional point $\boldsymbol{x}' \in [0,1]^n$ to be cut-off. The procedure returns a cut $\boldsymbol{\theta}^{*\top}\boldsymbol{x}' \leq 1 + \boldsymbol{\theta}^{*\top}\boldsymbol{u}$ whenever the optimal value of (12) is greater than one.

$$\max_{\boldsymbol{\omega},\boldsymbol{\theta}} \left\{ \boldsymbol{\theta}^\top(\boldsymbol{x}' - \boldsymbol{u}) :\ (11),\ \omega_{\mathbf{t}} = 0,\ \omega_{\mathbf{r}} = 1 + \boldsymbol{\theta}^\top \boldsymbol{u} \right\}. \tag{12}$$

Davarnia et al. (2020) [28] circumvent the need of an interior point by proposing a simpler but possibly non-linear BDD-based CGLP presented in (13). The model checks if any $\boldsymbol{x}'$ can be represented as a linear combination of points in X, i.e., whether there exists a $\boldsymbol{\theta}, \boldsymbol{\omega}$ such that $\boldsymbol{\theta}^\top \boldsymbol{x}' = \omega_{\mathbf{t}}$. Otherwise, their procedure returns a valid inequality $\boldsymbol{\theta}^{*\top} \boldsymbol{x} \leq \omega^*_{\mathbf{t}}$. Since the model may be unbounded, the optimization problem (13) includes normalization constraints $\mathcal{C}(\boldsymbol{\omega}, \boldsymbol{\theta}) \leq 0$ which are potentially non-linear. The CGLP (13) is addressed by an iterative subgradient separation algorithm.

$$\max_{\boldsymbol{\omega}, \boldsymbol{\theta}} \left\{ \boldsymbol{\theta}^\top \boldsymbol{x}' - \omega_{\mathbf{t}} : \ (11),\ \omega_{\mathbf{r}} = 0,\ \mathcal{C}(\boldsymbol{\omega}, \boldsymbol{\theta}) \leq 0 \right\}. \tag{13}$$

Note that, in our approach, we either solve BDD-CGLP (a linear program) or a single max-flow/min-cut problem, both relying only on $\mathcal{B}$.

6 Normally Distributed Linear Chance Constraints

For our numerical evaluation, we apply our combinatorial cut-and-lift procedure to binary problems with normally distributed chance constraints. Recall that problem CC considers a linear objective and m constraints of the form

$$\mathbb{P}(\boldsymbol{a}_j^\top \boldsymbol{x} \leq b_j) \geq \epsilon_j, \qquad \forall j \in \{1, ..., m\}, \tag{14}$$

where $\boldsymbol{a}_j \in \mathbb{R}^n$ is a random variable vector with joint normal distribution $N(\boldsymbol{\mu}_j, \Sigma_j^2)$, μ_{ji} is the mean for each a_{ji}, and $\Sigma_j^2 \in \mathbb{R}^{n \times n}$ is the covariance matrix of $\boldsymbol{a}_j$. Values $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{\epsilon} \in [0.5, 1]^m$ are deterministic parameters. Each constraint (14) can be written as a non-linear inequality $\boldsymbol{\mu}_j^\top \boldsymbol{x} + \Phi^{-1}(\epsilon_j) ||\Sigma_j \boldsymbol{x}||_2 \leq b_j$ for all $j \in \{1, ..., m\}$, where $\Phi(\cdot)$ is the cumulative distribution of a standard Gaussian and $||\cdot||_2$ is the Euclidean norm. In particular, this inequality is a special version of the second-order cone (SOC) constraint

$$\boldsymbol{\mu}^\top \boldsymbol{x} + \Omega ||\Sigma \boldsymbol{x} - \boldsymbol{d}||_2 \leq b \quad \Leftrightarrow \quad \boldsymbol{\mu}^\top \boldsymbol{x} + \Omega \sqrt{\sum_{k \in \{1, ..., l\}} (\boldsymbol{\sigma}_k^\top \boldsymbol{x} - d_k)^2} \leq b, \tag{15}$$

for a given vector $\boldsymbol{d} \in \mathbb{R}^l$, a positive constant $\Omega \in \mathbb{R}_+$, and matrix $\Sigma = [\boldsymbol{\sigma}_1, ..., \boldsymbol{\sigma}_l]^\top$ such that $\boldsymbol{\sigma}_k \in \mathbb{R}^n$ for all $k \in \{1, ..., l\}$.

We propose a novel BDD encoding for the general SOC inequalities (15) and evaluate its effectiveness for cases arising in normally distributed chance constraints. For additional experiments with classical cutting methods for SOC, we also present a BDD encoding for chance constraints with independent distributions, i.e., where $\Sigma_j^2 \in \mathbb{R}^{n \times n}$ is a diagonal matrix. In this case, inequality (15) reduces to a SOC knapsack [6,35] of the form

$$\boldsymbol{\mu}^\top \boldsymbol{x} + \Omega \sqrt{\sum_{i \in I} \sigma_{ii}^2 x_i} \leq b. \tag{16}$$

BDD encodings are built directly from a dynamic programming reformulation of the problem [15]. In our reformulation, we consider $l + 1$ sets of

state variables, $\boldsymbol{Q}_0, \boldsymbol{Q}_1, ..., \boldsymbol{Q}_l$, where each set of variables has $n+1$ stages, i.e, $\boldsymbol{Q}_k \in \mathbb{R}^{n+1}$ for each $k \in \{0, 1, ..., l\}$. State variables $\boldsymbol{Q}_0$ represent the value of the linear term (i.e., $\boldsymbol{\mu}^\top \boldsymbol{x}$), while $\boldsymbol{Q}_k$ encodes the k-th linear expression in the quadratic term (i.e, $\boldsymbol{\sigma}_k^\top \boldsymbol{x} - d_k$). Our recursive model for (15) is given by

$$
\begin{aligned}
\text{RSOC} := \{(\boldsymbol{x}; \boldsymbol{Q}) \in \{0,1\}^n \times \mathbb{R}^{(l+1)\times(n+1)} :& \\
Q_{0,0} = 0,\ Q_{k,0} = d_k, & \qquad \forall k \in \{1, ..., l\}, \quad \text{(17a)} \\
Q_{0,i} = Q_{0,i-1} + \mu_i x_i, & \qquad \forall i \in I, \quad \text{(17b)} \\
Q_{k,i} = Q_{k,i-1} + \sigma_{ki} x_i, & \qquad \forall i \in I,\ k \in \{1, ..., l\}, \quad \text{(17c)} \\
Q_{0,n} + \Omega \sqrt{\sum_{k=1}^{l} (Q_{k,n})^2} \leq b & \qquad \}. \quad \text{(17d)}
\end{aligned}
$$

The first set of equalities (17a) initialize the state variables at stage 0. Equalities (17b) and (17c) correspond to the recursive formulas for each of the linear expressions, and constraint (17d) enforces the SOC inequality. Notice that $\text{Proj}_{\text{x}}(RSOC)$ is equivalent to the feasible set of the SOC inequality (15).

For the case of the SOC knapsack inequality (16), the recursive model utilizes two sets of state variables, $\boldsymbol{Q}_0, \boldsymbol{Q}_1$, with $n+1$ stages each. As before, $\boldsymbol{Q}_0$ represents the value of the linear term and $\boldsymbol{Q}_1$ encodes the linear term inside the square root. Thus, the recursive model is

$$
\begin{aligned}
\text{RSOC\_KNAP} := \{(\boldsymbol{x}; \boldsymbol{Q}) \in \{0,1\}^n \times \mathbb{R}^{(l+1)\times(n+1)} :\ & \text{(17a)}, \text{(17b)}, \\
Q_{1,i} = Q_{1,i-1} + \sigma_{i,i}^2 x_i, & \qquad \forall i \in I, \\
Q_{0,n} + \Omega \sqrt{\sum_{i \in I} Q_{1,n}} \leq b & \qquad \}.
\end{aligned}
$$

An exact BDD $\mathcal{B} = (\mathcal{N}, \mathcal{A})$ is the reduced state-transition graph of the dynamic program above, where each BDD node maps to a state variable and each arc represents a state transition. If we consider, for example, the RSOC model, the root node $\mathbf{r}$ stores the stage-0 values, and each node u in layer $i \in I$ corresponds to a reachable state from the $(i-1)$-th stage. The recursions (17b) and (17c) are used to compute the transitions between nodes.

An exact BDD based on RSOC or RSOC_KNAP can be exponentially large. Therefore, we construct relaxed BDDs using a standard incremental refinement procedure [17]. For completeness, we provide a detailed explanation of our approach in Appendix A.

7 Empirical Evaluation and Discussion

This section presents an empirical evaluation of our combinatorial cut-and-lift procedure for CC (see §6). We create a BDD for each of the m chance constraints and apply our procedure for each such constraint at the root node of the branch-and-bound tree. For any fractional point $\boldsymbol{x} \in [0,1]^n$, we iterate

over each BDD until one of them generates either a general or combinatorial BDD flow cut, as we describe in detail below. We then lift the inequality using Algorithm 1. The procedure ends when $\boldsymbol{x}$ cannot be cut-off by any BDD.

We test our approach over the knapsack chance constraints (KCC) data set [6,35] with $n \in \{100, 125, 150\}$ variables and $m \in \{10, 20\}$ constraints. We also generate a random set of instances for general chance constraints (GCC) (i.e., inequality (15) with $\boldsymbol{d} = \boldsymbol{0}$) following a similar procedure to the one used for KCC. We consider $n \in \{75, 100, 125\}$, $m \in \{10, 20\}$, $\Omega \in \{1, 3, 5\}$, and a density of $2/\sqrt{n}$ over all the constraints. Parameters $\boldsymbol{\mu}_j$, Σ_j and $\boldsymbol{c}$ are sampled from a discrete uniform distribution with $\boldsymbol{\mu}_j \in [-50, 50]^n$, $\Sigma_j \in [-20, 20]^{n \times n}$ and $\boldsymbol{c} \in [0, 100]^n$. Parameters b_j are given by

$$b_j = t \cdot \left(\sum_{k=1}^{n} \mu_{jk}^{+} + \Omega \sqrt{\sum_{i=1}^{n} \max \left\{ \sum_{k=1}^{n} \sigma_{jik}^{+}, \sum_{k=1}^{n} \sigma_{jik}^{-} \right\}^2} \right), \qquad \forall j \in \{1, ..., m\},$$

where $t \in \{0.1, 0.2, 0.3\}$ is the constraint tightness, $a^+ := \max\{0, a\}$, and $a^- := \max\{0, -a\}$ for any $a \in \mathbb{R}$. Notice that b_j with $t = 0.3$ will remove approximately 50% of the possible assignments for $\boldsymbol{x} \in \{0, 1\}^n$. Then, we generate 5 random instances for each combination of n, m, Ω, and t, i.e., a total of 270 instances.

We implement four variants of our approach to test the effectiveness of the BDD cuts and lifting procedure. The first two, BW and BWL, apply the weaker combinatorial BDD cutting-plane procedure (see §5.3), where BW omits the lifting procedure and BWL includes it. The other two variants, BG and BGL, use the weaker BDD flow cuts first and try the general BDD flow cuts (see §5.2) if the weaker approach fails to produce a cut. As before, BGL utilizes our lifting procedure in every generated constraint while BG does not.

We also implement the cover cuts and lifting procedure by Atamtürk and Narayanan (2009) [6] for the KCC case. We test their cover cuts with and without their continuous SOC lifting, C and CL, respectively. In addition, we experiment using their cover cuts in conjunction with our BDD lifting, BCL.

Our procedures are implemented in `C++` in the IBM ILOG CPLEX 12.9 solver using the `UserCuts` callback at the root node of the search.[1] All experiments consider a single thread, a one-hour time limit, and the linearization strategy (i.e., `MIQCPStrat = 2`) to solve the SOC problems.[2] We deactivate all solver cuts when running our techniques (i.e., BW, BWL, BG, and BGL) and the cover cut variants (i.e., C, CL, and BCL) to evaluate their effectiveness on their own. Notice that, given the `UserCuts` callback, our techniques omit the `Presolve` option and use `TraditionalSearch`. We use the same configuration when running CPLEX to have a fair comparison.[3]

[1] We will make the code available upon publication.

[2] Preliminary experiments show that this was the best strategy across all techniques.

[3] Preliminary results show, in fact, that CPLEX performs better on the problem sets when `Presolve` is *deactivated*.

Table 1 Aggregated results showing the overall performance of each technique for GCC.

	# Solve	Root Gap	Final Gap	Time (sec)	# Nodes	# Cuts	% Lifted
CPLEX	137	20.8%	7.7%	550.4	176,860.3	128	-
BW	139	20.0%	7.3%	409.5	252,865.5	31	-
BWL	150	15.7%	5.9%	280.8	150,200.1	23	90.7%
BG	166	13.5%	4.6%	210.0	84,592.0	324	-
BGL	**168**	**13.4%**	**4.4%**	**169.7**	**78,058.3**	132	72.0%

We tested with three maximum BDD widths $\mathcal{W} = \{2000, 3000, 4000\}$, i.e, the maximum number of nodes per layer allowed in a relaxed BDD. We present results for the width with the best overall performance, $\mathcal{W} = 4000$ (we include detailed results for other widths in Appendix B). Notice that most of the created BDDs are relaxed due to width limit, especially when $n \geq 100$.

7.1 Overall Performance and Comparison

Tables 1 and 2 present the average results of all techniques for the 270 GCC instances and the 90 KCC instances, respectively. The first column presents the number of problems solved to optimality. The second and third columns correspond to the average root gap and final gap across all instances, i.e., $gap = (UB - LB)/LB$, with UB the root/final upper bound and LB the best lower bound found by all techniques. The fourth and fifth columns present the average solving time (including the BDD construction time) and nodes explored for the subset of instances that all techniques solve. The sixth column shows the number of cuts added by either the solver (i.e., for CPLEX) or our techniques. The last column presents the average percentage of original constraints that are lifted at least one time. See Appendix C and D for detailed results over each configuration.

Table 1 shows that all our variants have better performance than CPLEX. Specifically, BGL has the best performance solving 31 more instances than CPLEX. The difference on instances solved can be explained by the root node gap reduction for our approaches. Also, our lifting variants consistently solve more instances and are on average faster than BW and BG.

Table 2 Aggregated results showing the overall performance of each technique for KCC.

	# Solve	Root Gap	Final Gap	Time (sec)	# Nodes	# Cuts	% Lifted
CPLEX	70	2.84%	0.39%	174.7	161,210.1	123.5	-
C	71	3.29%	0.41%	233.4	489,853.8	96.4	-
CL	70	2.81%	0.34%	153.2	306,060.5	95.7	98.6%
BCL	70	2.67%	0.27%	**117.8**	199,002.4	81.0	70.0%
BW	64	3.63%	0.53%	500.1	938,585.4	240.6	-
BWL	74	2.80%	0.26%	161.6	310,699.6	54.9	99.2%
BG	76	**1.34%**	0.16%	468.7	48,056.1	1087.0	-
BGL	**88**	**1.33%**	**0.01%**	139.0	**35,661.5**	192.4	80.5%

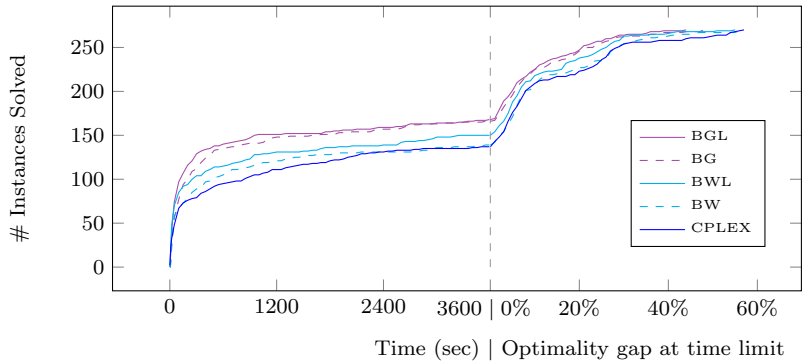

Fig. 2 Profile plot comparing the accumulated number of instances solved over time (left), and the accumulated number of instances over a final gap range (right) for GCC dataset.

The average run time for instances solved is significantly lower for our techniques (see Table 1). However, the number of nodes explored does not necessarily correlate with shorter solving time when looking at the disaggregated results (see Appendix D). We believe that this is influenced by the number of cuts added at the root node and the resulting root gap.

Similarly, Table 2 shows that the general BDD flow cuts (i.e. BG, and BGL) solves up to 17 more instances than the best baselines. Moreover, BGL has a 52.5% gap reduction when compared to CL and an order of magnitude reduction on nodes explored. Also, our lifting procedure slightly outperforms the continuous SOC lifting [6]; BCL achieves a smaller average root and final gap than BCL, in addition to decreasing the run time and nodes explored. Lastly, BW has a significant performance improvement when enhanced by our lifting procedure, BWL, while there is a marginal improvement for lifting cover cuts (i.e., BCL and CL).

Figure 2 depicts the performance of each algorithm for the GCC instances (similar results can be found for KCC in Appendix C). The graph illustrates the number of instances solved over time (left side) and the accumulated number of instances over a final gap range (right side). We see a clear dominance of our general BDD flow cuts (i.e., BG and BGL) and also the impact of lifting in number of instances solved and gap reduction. In particular, BG and BGL have the largest gap reductions and BWL has a consistently smaller gap than CPLEX. BW also improves upon CPLEX by a small margin.

While the results show that our techniques are on average superior to CPLEX, we uncovered problem characteristics where our best approach, BGL, performs significantly better. Figure 3 shows two plots comparing the root gap of BGL and CPLEX for the GCC instances and different values of Ω and t. In each plot, an (x, y) point represents the root gap for an instance given by the x-axis and the y-axis technique, respectively. Overall, we can see that BGL achieves a smaller or equal root gap to CPLEX, however, the difference is considerably larger when $\Omega \geq 3$ and $t = 0.1$.

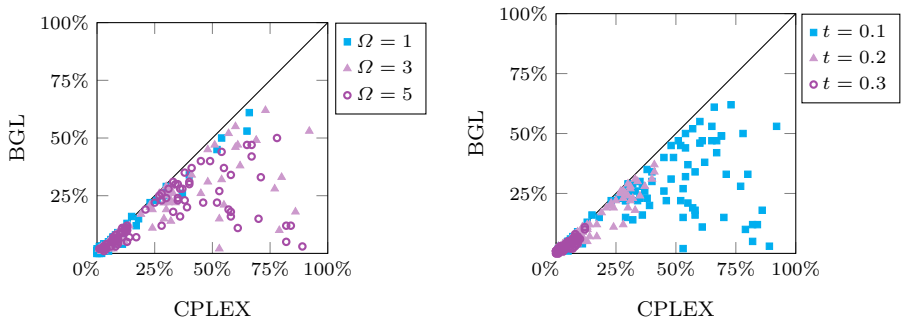

Fig. 3 Root node gap comparison with CPLEX over the GCC instances. The right plot illustrate the gap difference for different Ω values and the left plot for different t values.

The problems become more challenging with a larger Ω (i.e., the quadratic term is more predominant) because the SOC relaxation and its linearization are quite weak. Thus, our combinatorial cut-and-lift can potentially generate stronger cuts than CPLEX. In fact, the left plot in Figure 3 shows all instances with $\Omega = 1$ close to the diagonal, while problems with $\Omega \in \{3, 5\}$ have larger gap reductions. Lastly, the right plot of Figure 3 shows that BGL has significantly smaller gaps than CPLEX over instances with a smaller t (i.e., smaller solution sets per constraint). Our relaxed BDDs were closer to exact BDDs in these cases, thus, making our cuts more effective.

7.2 Effectiveness of BDD-based Cutting and Lifting Procedure

We now evaluate the effectiveness of our four variants by comparing their optimality gaps at the root node. We focus on the GCC instances since we observed a similar behavior over the KCC dataset (see Appendix C).

Figure 4 depicts three plots making a pairwise comparison of the root gap for all variants. The left plot compares the root gap of BW and BG, where BG achieves equal or better gaps than BW since most points lie on or below the diagonal. Recall that the weaker combinatorial BDD flow cuts are incomplete, thus, it is expected that BG has smaller gaps than BW. The gap difference translates into 27 more instances solved by BG than BW (see Table 1).

The last two plots in Figure 4 illustrate the effectiveness of the lifting procedure when using the combinatorialx (middle) and the general (right) BDD flow cuts. We can see that BWL has smaller gaps than BW and its average root gap is closer BG than to BW (see Table 1). However, there is no clear root gap improvement when using the lifting procedure over the general BDD cuts. This is not a surprise since BG is able to separate all points outside each BDD solution set. Nonetheless, we can see the effectiveness of our lifting procedure later on, where BGL solves 2 instances more than BG, adds fewer cuts, and has the smallest average final gaps across all variants (see Table 1).

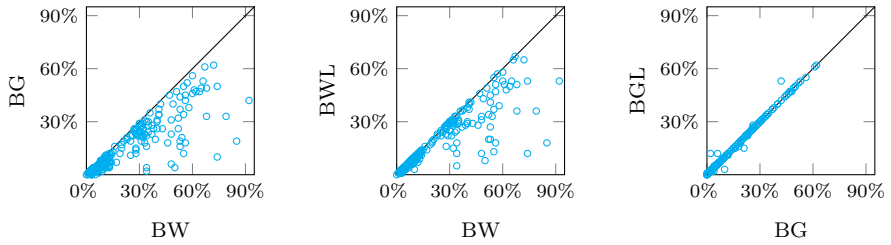

Fig. 4 Pairwise root gap comparison for our four variants over the GCC instances: BW vs. BG (left), BW vs. BWL (middle), and BG vs. BGL (right).

8 Conclusions

We introduce a novel lifting and cutting-plane procedure for binary programs that leverage their combinatorial structure via a binary decision diagram (BDD) encoding of their constraints. Our lifting procedure relies on 0-1 disjunctions to rotate valid inequalities and uses a BDD to efficiently compute the disjunctive sub-problems. While our combinatorial lifting can enhance any cutting-plane approach, we also propose a new BDD-based cut generation algorithm based on an alternative network-flow representation of the BDD.

BDDs give us the flexibility to apply our procedure to a wide range of nonlinear problems. As a study case, we tested our procedure over normally distributed linear chance-constrained problems, a common application of second-order cone (SOC) inequalities. To do so, we introduce a novel BDD encoding for these constraints and compare the performance of our procedure against a state-of-the-art solver and existing cut-and-lift procedure for SOC knapsacks. The empirical results show that our technique solves 31 more instances, reducing the final gap by 42.6% and having a threefold decrease in run-time over the general chance-constrained instances. In addition, our technique outperforms existing cut-and-lift methodologies for SOC knapsack problems by solving 17 more instances, achieving a 96.3% final gap reduction, and having comparable average solving time. We note that our procedure performs best when the solution set of each inequality is smaller and the quadratic term of the SOC inequalities is predominant.

References

1. Ahmed, S., Luedtke, J., Song, Y., Xie, W.: Nonanticipative duality, relaxations, and formulations for chance-constrained stochastic programs. Mathematical Programming **162**(1-2), 51–81 (2017)
2. Ahmed, S., Shapiro, A.: Solving chance-constrained stochastic programs via sampling and integer programming. In: State-of-the-art decision-making tools in the information-intensive age, pp. 261–269. Informs (2008)
3. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Inc., USA (1993)

4. Andersen, H.R., Hadzic, T., Hooker, J.N., Tiedemann, P.: A constraint store based on multivalued decision diagrams. In: International Conference on Principles and Practice of Constraint Programming–CP 2007, pp. 118–132. Springer (2007)
5. Atamtürk, A., Bhardwaj, A.: Network design with probabilistic capacities. Networks **71**(1), 16–30 (2018)
6. Atamtürk, A., Narayanan, V.: The submodular knapsack polytope. Discrete Optimization **6**(4), 333–344 (2009)
7. Atamtürk, A., Narayanan, V.: Conic mixed-integer rounding cuts. Mathematical programming **122**(1), 1–20 (2010)
8. Balas, E.: Facets of the knapsack polytope. Mathematical programming **8**(1), 146–164 (1975)
9. Balas, E.: Disjunctive Programming. Springer (2018)
10. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0–1 programs. Mathematical programming **58**(1-3), 295–324 (1993)
11. Balas, E., Ceria, S., Cornuéjols, G.: Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. Management Science **42**(9), 1229–1246 (1996)
12. Becker, B., Behle, M., Eisenbrand, F., Wimmer, R.: BDDs in a branch and cut framework. In: International Workshop on Experimental and Efficient Algorithms, pp. 452–463. Springer (2005)
13. Behle, M.: Binary decision diagrams and integer programming. Ph.D. Thesis (2007)
14. Bergman, D., Cardonha, C., Mehrani, S.: Binary decision diagrams for bin packing with minimum color fragmentation. In: International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research–CPAIOR 2019, pp. 57–66. Springer (2019)
15. Bergman, D., Cire, A.A.: Discrete nonlinear optimization by state-space decompositions. Management Science **64**(10), 4700–4720 (2018)
16. Bergman, D., Cire, A.A., van Hoeve, W.J., Hooker, J.N.: Variable ordering for the application of BDDs to the maximum independent set problem. In: International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research–CPAIOR 2012, pp. 34–49. Springer (2012)
17. Bergman, D., Cire, A.A., van Hoeve, W.J., Hooker, J.N.: Discrete optimization with decision diagrams. INFORMS Journal on Computing **28**(1), 47–66 (2016)
18. Bergman, D., Lozano, L.: Decision diagram decomposition for quadratically constrained binary optimization. Optimization Online e-prints (2018)
19. Bixby, R.E., Fenelon, M., Gu, Z., Rothberg, E., Wunderling, R.: Mixed-integer programming: A progress report. In: The sharpest cut: the impact of Manfred Padberg and his work, pp. 309–325. SIAM (2004)
20. van den Bogaerdt, P., de Weerdt, M.: Multi-machine scheduling lower bounds using decision diagrams. Operations Research Letters **46**(6), 616–621 (2018)
21. van den Bogaerdt, P., de Weerdt, M.: Lower bounds for uniform machine scheduling using decision diagrams. In: International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research–CPAIOR 2019, pp. 565–580. Springer (2019)
22. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. Computers, IEEE Transactions on **100**(8), 677–691 (1986)
23. Castro, M.P., Cire, A.A., Beck, J.C.: An MDD-based lagrangian approach to the multicommodity pickup-and-delivery tsp. INFORMS Journal on Computing (2019)
24. Castro, M.P., Piacentini, C., Cire, A.A., Beck, J.C.: Relaxed BDDs: An admissible heuristic for delete-free planning based on a discrete relaxation. In: Proceedings of the International Conference on Automated Planning and Scheduling, pp. 77–85 (2019)
25. Charnes, A., Cooper, W.W.: Chance-constrained programming. Management science **6**(1), 73–79 (1959)
26. Cire, A.A., van Hoeve, W.J.: Multivalued decision diagrams for sequencing problems. Operations Research **61**(6), 1411–1428 (2013)
27. Cohen, M.C., Keller, P.W., Mirrokni, V., Zadimoghaddam, M.: Overcommitment in cloud services: Bin packing with chance constraints. Management Science **65**(7), 3255–3271 (2019)
28. Davarnia, D., van Hoeve, W.J.: Outer approximation for integer nonlinear programs via decision diagrams. Mathematical Programming (2020)

29. Gomory, R.E.: Some polyhedra related to combinatorial problems. Linear Algebra and its Applications **2**(4), 451 – 558 (1969)
30. Gurobi Optimization, L.: Gurobi optimizer reference manual (2020)
31. Hammer, P.L., Johnson, E.L., Peled, U.N.: Facet of regular 0–1 polytopes. Mathematical Programming **8**(1), 179–206 (1975)
32. Hoda, S., Van Hoeve, W.J., Hooker, J.N.: A systematic approach to MDD-based constraint programming. In: International Conference on Principles and Practice of Constraint Programming–CP 2010, pp. 266–280. Springer (2010)
33. Hooker, J.N.: Job sequencing bounds from decision diagrams. In: International Conference on Principles and Practice of Constraint Programming–CP 2017, pp. 565–578. Springer (2017)
34. IBM: ILOG CPLEX Studio 12.9 Manual (2019)
35. Joung, S., Park, S.: Lifting of probabilistic cover inequalities. Operations Research Letters **45**(5), 513–518 (2017)
36. Kinable, J., Cire, A.A., van Hoeve, W.J.: Hybrid optimization methods for time-dependent sequencing problems. European Journal of Operational Research **259**(3), 887–897 (2017)
37. Lodi, A.: Mixed integer programming computation. In: 50 Years of Integer Programming 1958-2008, pp. 619–645. Springer (2010)
38. Lodi, A., Tanneau, M., Vielma, J.P.: Disjunctive cuts for mixed-integer conic optimization. arXiv preprint arXiv:1912.03166 (2019)
39. Louveaux, Q., Wolsey, L.A.: Lifting, superadditivity, mixed integer rounding and single node flow sets revisited. Quarterly Journal of the Belgian, French and Italian Operations Research Societies **1**(3), 173–207 (2003)
40. Lozano, L., Smith, J.C.: A binary decision diagram based algorithm for solving a class of binary two-stage stochastic programs. Mathematical Programming pp. 1–24 (2018)
41. Luedtke, J., Ahmed, S.: A sample approximation approach for optimization with probabilistic constraints. SIAM Journal on Optimization **19**(2), 674–699 (2008)
42. Luedtke, J., Ahmed, S., Nemhauser, G.L.: An integer programming approach for linear programs with probabilistic constraints. Mathematical programming **122**(2), 247–272 (2010)
43. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley-Interscience, USA (1988)
44. Padberg, M.W.: On the facial structure of set packing polyhedra. Mathematical programming **5**(1), 199–215 (1973)
45. Padberg, M.W.: A note on zero-one programming. Operations Research **23**(4), 833–837 (1975)
46. Pagnoncelli, B.K., Ahmed, S., Shapiro, A.: Sample average approximation method for chance constrained programming: theory and applications. Journal of optimization theory and applications **142**(2), 399–416 (2009)
47. Van de Panne, C., Popp, W.: Minimum-cost cattle feed under probabilistic protein constraints. Management Science **9**(3), 405–430 (1963)
48. Perregaard, M., Balas, E.: Generating cuts from multiple-term disjunctions. In: International Conference on Integer Programming and Combinatorial Optimization, pp. 348–360. Springer (2001)
49. Raghunathan, A.U., Bergman, D., Hooker, J.N., Serra, T., Kobori, S.: Seamless multimodal transportation scheduling. arXiv preprint arXiv:1807.09676 (2018)
50. Shylo, O.V., Prokopyev, O.A., Schaefer, A.J.: Stochastic operating room scheduling for high-volume specialties under block booking. INFORMS Journal on Computing **25**(4), 682–692 (2013)
51. Tjandraatmadja, C., van Hoeve, W.J.: Target cuts from relaxed decision diagrams. INFORMS Journal on Computing **31**(2), 285–301 (2019)
52. Vielma, J.P., Ahmed, S., Nemhauser, G.L.: A lifted linear programming branch-and-bound algorithm for mixed-integer conic quadratic programs. INFORMS Journal on Computing **20**(3), 438–450 (2008)
53. Vielma, J.P., Dunning, I., Huchette, J., Lubin, M.: Extended formulations in mixed integer conic quadratic programming. Mathematical Programming Computation **9**(3), 369–418 (2017)
54. Wolsey, L.A.: Technical notefacets and strong valid inequalities for integer programs. Operations Research **24**(2), 367–372 (1976). DOI 10.1287/opre.24.2.367

A Relaxed BDD Construction Procedure for Second-Order Cones

We now present the relaxed BDD construction procedure for chance constraints based on the RSOC model. The construction algorithm is analogous for the case of SOC knapsacks and the RSOC_KNAP model.

The state information at the core of our BDD construction algorithm is based on recursive model RSOC. This information is stored in each node of the BDD and is used to identify infeasible assignments and decide how to widen the BDD (i.e., split nodes). Our state information keeps track of each of the $l+1$ linear components in inequality (15), i.e., $\boldsymbol{\mu}^\top \boldsymbol{x}$ and $\boldsymbol{\sigma}_k^\top \boldsymbol{x} - d_k$ for each $k \in \{1, ..., l\}$. Thus, each node $u \in \mathcal{N}_i$ has two $l+1$ dimensional vectors for top-down information, $Q^{\downarrow \mathsf{min}}(u)$ and $Q^{\downarrow \mathsf{max}}(u)$, that approximate the linear components considering the partial assignments from $\mathbf{r}$ to u. We set the top-down state information at the root node as $Q_0^{\downarrow \mathsf{min}}(\mathbf{r}) := Q_0^{\downarrow \mathsf{max}}(\mathbf{r}) := 0$ and $Q_k^{\downarrow \mathsf{min}}(\mathbf{r}) := Q_k^{\downarrow \mathsf{max}}(\mathbf{r}) := -d_k$ for all $k \in \{1, ..., l\}$. Then, for any $u \in \mathcal{N}_i$, $i \in \{2, ..., n\}$, and $k \in \{1, ..., l\}$ we update the states as:

$$Q_0^{\downarrow \mathsf{min}}(u) := \min_{a \in \mathcal{A}^{\mathsf{in}}(u)} \{Q_0^{\downarrow \mathsf{min}}(s(a)) + \mu_{i-1} \cdot v_a\},$$
$$Q_0^{\downarrow \mathsf{max}}(u) := \max_{a \in \mathcal{A}^{\mathsf{in}}(u)} \{Q_0^{\downarrow \mathsf{max}}(s(a)) + \mu_{i-1} \cdot v_a\},$$
$$Q_k^{\downarrow \mathsf{min}}(u) := \min_{a \in \mathcal{A}^{\mathsf{in}}(u)} \{Q_k^{\downarrow \mathsf{min}}(s(a)) + \sigma_{ki-1} \cdot v_a\},$$
$$Q_k^{\downarrow \mathsf{max}}(u) := \max_{a \in \mathcal{A}^{\mathsf{in}}(u)} \{Q_k^{\downarrow \mathsf{max}}(s(a)) + \sigma_{ki-1} \cdot v_a\},$$

Similarly, we use two $l+1$ dimensional vectors, $Q^{\uparrow \mathsf{min}}(u)$ and $Q^{\uparrow \mathsf{max}}(u)$, for our bottom-up state information for each node $u \in \mathcal{N}$. The information is initialized at the terminal node as $Q_0^{\uparrow \mathsf{min}}(\mathbf{t}) := Q_0^{\uparrow \mathsf{max}}(\mathbf{t}) := 0$ and $Q_k^{\uparrow \mathsf{min}}(\mathbf{t}) := Q_k^{\uparrow \mathsf{max}}(\mathbf{t}) := 0$ for all $k \in \{1, ..., l\}$. Then, for any $u \in \mathcal{N}_i$, $i \in \{1, ..., n-1\}$, and $k \in \{1, ..., l\}$, we have:

$$Q_0^{\uparrow \mathsf{min}}(u) := \min_{a \in \mathcal{A}^{\mathsf{out}}(u)} \{Q_0^{\uparrow \mathsf{min}}(t(a)) + \mu_k \cdot v_a\},$$
$$Q_0^{\uparrow \mathsf{max}}(u) := \max_{a \in \mathcal{A}^{\mathsf{out}}(u)} \{Q_0^{\uparrow \mathsf{max}}(t(a)) + \mu_k \cdot v_a\},$$
$$Q_k^{\uparrow \mathsf{min}}(u) := \min_{a \in \mathcal{A}^{\mathsf{out}}(u)} \{Q_k^{\uparrow \mathsf{min}}(t(a)) + \sigma_{ki} \cdot v_a\},$$
$$Q_k^{\uparrow \mathsf{max}}(u) := \max_{a \in \mathcal{A}^{\mathsf{out}}(u)} \{Q_k^{\uparrow \mathsf{max}}(t(a)) + \sigma_{ki} \cdot v_a\}.$$

For each node $u \in \mathcal{N}$, the state information under and over approximates the value of the linear components of (15) for all $\mathbf{r} - u$ paths (i.e., top-down information) and for all $u - \mathbf{t}$ paths (i.e., bottom-up information). We used the state information to identify if an arc corresponds to an infeasible assignment, i.e., all paths traversing it correspond to infeasible solutions of (15). In particular, we can remove an arc $a = (u, u') \in \mathcal{A}_i$ if the following condition holds:

$$Q_0^{\downarrow \mathsf{min}}(u) + \mu_i \cdot v_a + Q_0^{\uparrow \mathsf{min}}(u') + \Omega \sqrt{\sum_{k=1}^{l} g_k(a)} > b, \tag{18}$$

where $g_k(a)$ for $k \in \{1, ..., l\}$ is given by:

$$g_k(a) := \begin{cases} (Q_k^{\downarrow \mathsf{min}}(u) + \sigma_{ki} \cdot v_a + Q_k^{\uparrow \mathsf{min}}(u'))^2, & \text{if } Q_k^{\downarrow \mathsf{min}}(u) + \sigma_{ki} \cdot v_a + Q_k^{\uparrow \mathsf{min}}(u') > 0, \\ (Q_k^{\downarrow \mathsf{max}}(u) + \sigma_{ki} \cdot v_a + Q_k^{\uparrow \mathsf{max}}(u'))^2, & \text{if } Q_k^{\downarrow \mathsf{max}}(u) + \sigma_{ki} \cdot v_a + Q_k^{\uparrow \mathsf{max}}(u') < 0, \\ 0, & \text{otherwise.} \end{cases}$$

Notice that $g_k(a)$ under approximates $(\boldsymbol{\sigma}_k^\top \boldsymbol{x} - d_k)^2$ for all paths traversing arc $a \in \mathcal{A}$, and so the left-hand side (LHS) of (18) under approximates the LHS of (15) for all paths

traversing arc $a \in \mathcal{A}_i$. Then, all paths traversing an arc a that satisfy (18) correspond to invalid assignments for (15).

If $Q_k^{\downarrow\mathsf{max}}(u) = Q_k^{\downarrow\mathsf{min}}(u)$ for all nodes $u \in \mathcal{N}$, we recover the exact BDD based on the recursive model RSOC and condition (18) is equivalent to (17d). Thus, our splitting procedure tries to achieve this property by selecting nodes u with $Q_k^{\downarrow\mathsf{max}}(u) - Q_k^{\downarrow\mathsf{min}}(u) \geq \delta$ ($\delta > 0$) for some $k \in \{0, ..., l\}$ and then split it into two new nodes, u' and u'', so $Q_k^{\downarrow\mathsf{max}}(u') - Q_k^{\downarrow\mathsf{min}}(u') < \delta$ and $Q_k^{\downarrow\mathsf{max}}(u'') - Q_k^{\downarrow\mathsf{min}}(u'') < \delta$. The splitting procedure then duplicates the outgoing arcs of u and assigns them to both u' and u'' to keep the same set of paths in $\mathcal{B}$.

Algorithm 2 Relaxed (Exact) BDD Construction Procedure

```
1:  procedure ConstructBDD(SOC Constraint ⟨μ, Σ, d, Ω, b, n⟩, W)
2:      B := WidthOneBDD(n)
3:      while B has been modified do
4:          UpdateBDDNodesBottom(N)
5:          for i ∈ {1, ..., n} do
6:              UpdateBDDNodesTop(N_i)
7:              SplitBDDNodes(N_i, W)
8:              FilterBDDOutgoingEdges(N_i)
9:          UpdateBDDNodesTop(N_{n+1})
10:     ReduceBDD(B)
11:     return B
```

Our construction procedure creates a relaxed BDD $\mathcal{B} = (\mathcal{N}, \mathcal{A})$ by limiting its width $w(\mathcal{B})$ by a positive value $\mathcal{W}$, where $w(\mathcal{B}) := \max_{i \in I}\{|N_i|\}$ represents the maximum number of nodes in each layer. The complete BDD construction procedure is shown in Algorithm 2. The algorithm starts creating a width-one BDD for the SOC constraint (line 2) and then updates the bottom-up information for all the nodes (line 4). During the top-down pass through the BDD (lines 5-9), the procedure updates the top-down information of layer $\mathcal{N}_i$, splits the nodes until we reach the width limit $\mathcal{W}$, and filters the emanating arcs of $\mathcal{N}_i$. The algorithm then checks if the BDD has been updated (line 3) and repeats the bottom-up and top-down iterations until the BDD cannot be updated any more. Lastly, we reduce the BDD (line 10) following the standard procedure in the literature [22].

The resulting BDD starts with all possible variable assignments (i.e., a width-one BDD) and removes arcs using condition (18). Thus, the procedure is guaranteed to construct a relaxed BDD for a SOC constraint. Notice that for a big enough $\mathcal{W}$, the procedure will return an exact BDD.

Example 5 Consider the following binary set with an SOC constraints $X = \{\boldsymbol{x} \in \{0,1\}^3 : 3x_1 + x_2 + x_3 + \sqrt{(x_1 + x_2 + 2x_3)^2 + (x_1 + 3x_2 - x_3 + 3)^2} \leq 8\}$. Figure 5 depicts some of the steps to construct an exact BDD for X. The left most diagram corresponds to a width-one BDD for this problem. The top-down state information in the root node is $((Q_0^{\downarrow\mathsf{min}}(\mathbf{r}), Q_0^{\downarrow\mathsf{max}}(\mathbf{r})),\ (Q_1^{\downarrow\mathsf{min}}(\mathbf{r}), Q_1^{\downarrow\mathsf{max}}(\mathbf{r})),\ (Q_2^{\downarrow\mathsf{min}}(\mathbf{r}), Q_2^{\downarrow\mathsf{max}}(\mathbf{r}))) = ((0,0),\ (0,0),\ (3,3))$, while for node u_1 is $((0,3),\ (0,1),\ (3,4))$.

The middle BDD illustrates the resulting BDD after splitting node u_1. The resulting nodes, u_1' and u_1'', have top-down state information $((0,0),\ (0,0),\ (3,3))$ and $((3,3),\ (1,1),\ (4,4))$, respectively. In addition, the gray arc from u_1'' to u_2 corresponds to an invalid assignment: the bottom-up information of u_2 is $((0,1),\ (0,2),\ (-1,0))$, thus, (18) evaluates to $10.3 > 8$.

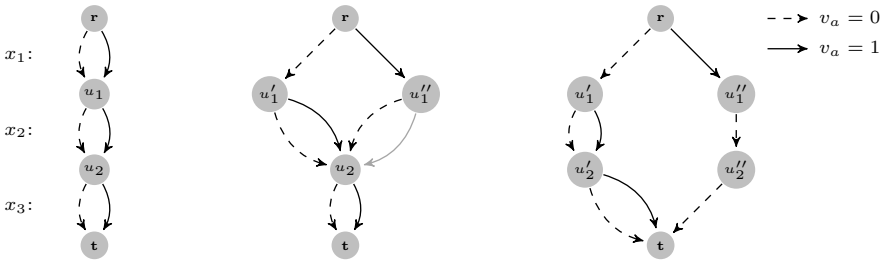

Fig. 5 BDD construction procedure for set X defined in Example 5. The figure depicts a width-one BDD (left), a BDD after the splitting and filtering procedure over $\mathcal{N}_2$ (middle), and the resulting exact reduced BDD (right).

B Experiments Comparing Different BDD Widths

Table 3 presents the average performance for CPLEX and our four alternatives (i.e, BW, BWL, BG, and BGL) with three different maximum width values, $\mathcal{W} \in \{2000, 3000, 4000\}$, over the GCC instances. The table shows the number of instances solved, average root gap, and average final gap for all techniques. Our four alternatives with $\mathcal{W} \in \{2000, 3000, 4000\}$ each outperform CPLEX. $\mathcal{W} = 4000$ achieves the best overall performance across the four combinatorial cut-and-lift alternatives.

Table 3 Average performance of all techniques for different BDD widths for GCC.

	Width	# Solve	Root Gap	Final Gap
CPLEX		137	20.82%	7.75%
BW	2000	137	20.31%	7.35%
	3000	**140**	20.11%	**7.25%**
	4000	139	**20.01%**	**7.25%**
BWL	2000	149	16.61%	6.09%
	3000	**150**	16.03%	6.04%
	4000	**150**	**15.68%**	**5.89%**
BG	2000	160	14.93%	5.21%
	3000	159	14.05%	4.87%
	4000	**166**	**13.47%**	**4.59%**
BGL	2000	160	14.91%	5.00%
	3000	**168**	14.03%	4.66%
	4000	**168**	**13.44%**	**4.43%**

Similarly, Table 4 presents the average performance over the KCC instances for CPLEX, our four alternatives (i.e, BW, BWL, BG, and BGL), and cover cuts with BDD lifting (i.e., BCL) with three different maximum width values, $\mathcal{W} \in \{2000, 3000, 4000\}$. Overall, $\mathcal{W} = 4000$ achieves the best or comparable performance across the five BDD approaches.

Table 4 Average performance of all techniques for different BDD widths for KCC.

	Width	# Solve	Root Gap	Final Gap
CPLEX		70	2.84%	0.39%
BCL	2000	**71**	2.72%	0.29%
	3000	70	2.69%	0.28%
	4000	70	**2.67%**	**0.27%**
BW	2000	66	3.64%	**0.52%**
	3000	**67**	3.63%	**0.52%**
	4000	64	**3.63%**	0.53%
BWL	2000	74	2.84%	0.27%
	3000	72	**2.80%**	0.28%
	4000	**74**	**2.80%**	**0.26%**
BG	2000	75	1.62%	0.16%
	3000	**78**	1.45%	**0.15%**
	4000	76	**1.34%**	0.16%
BGL	2000	83	1.61%	0.04%
	3000	87	1.44%	0.02%
	4000	**88**	**1.33%**	**0.01%**

C Average Performance Comparison for Knapsack Chance Constraints

We now present additional results for the KCC dataset. Figure 6 compares our best variant, BGL, against CPLEX (left plot) and CL (right plot). In both cases, we see a clear superiority of our combinatorial cut-and-lift approach, especially when $\Omega \geq 3$.

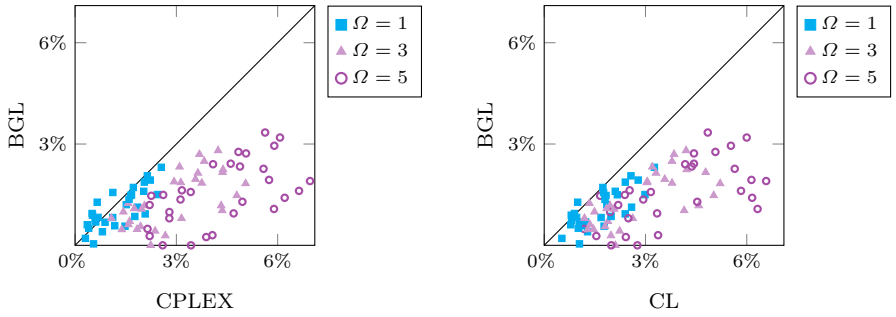

Fig. 6 Root gap comparison between our best variant BGL, CPLEX, and CL.

As in Figure 2, Figure 7 illustrate the performance of each algorithm for the KCC dataset. We see a clear dominance of BGL and also the positive impact of our combinatorial lifting in instances solved and gap reduction.

Figure 8 presents an instance-wise root gap comparison for our four BDD variants. We observed a similar behavior to the results for the GCC instances (see Figure 4). Similarly, Figure 9 compares our combinatorial cut-and-lift to the cover cut alternatives. We can see that our combinatorial BDD flow cuts perform worse than the cover cuts (left plot), while the techniques are comparable when enhanced by their respective lifting procedures (right

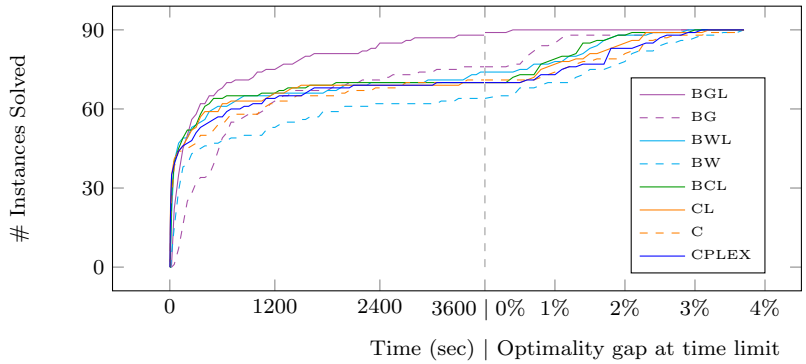

Fig. 7 Profile plot comparing the accumulated number of instances solved over time (left), and the accumulated number of instances over a final gap range (right) for KCC dataset.

plot). Lastly, our BDD lifting is slightly better than the continuous lifting over the cover cuts (middle plot).

Table 5 shows the number of instances solved, average root gap, and average final gap for each n, m, and Ω combination with $\mathcal{W} = 4000$. Similarly, Table 6 shows the average number of nodes in the branch-and-bound search and the average run time for the instances that all techniques solved to optimality. Column # shows the number of instances used to compute the average results.

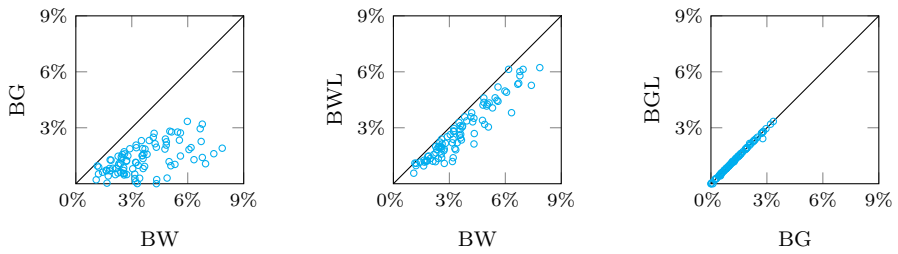

Fig. 8 Root gap comparison between our techniques for KCC dataset.

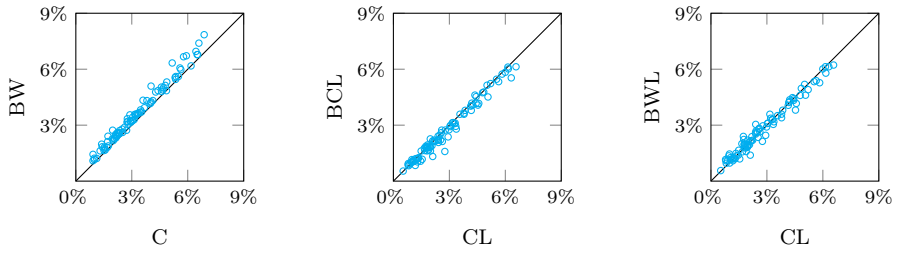

Fig. 9 Root gap comparison: Our techniques vs. existing approaches for KCC instances.

Table 5 Gap comparison across all techniques for KCC instances.

			# Instances Solved to Optimality								Av. Root Gap (%)								Av. Final Gap (%)							
n	m	Ω	CPLEX	C	CL	BCl	BW	BWL	BG	BGL	CPLEX	C	CL	BCl	BW	BWL	BG	BGL	CPLEX	C	CL	BCl	BW	BWL	BG	BGL
100	10	1	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	0.9	1.8	1.4	1.4	2.0	1.4	**0.5**	**0.5**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		3	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	2.4	2.9	2.2	1.9	3.2	2.0	0.5	**0.4**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		5	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	3.7	3.9	2.8	2.2	4.4	2.6	**0.3**	**0.3**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	20	1	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	1.9	2.8	2.4	2.3	3.2	2.5	**1.1**	**1.1**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		3	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	4.5	5.0	4.5	4.4	5.4	4.3	**1.5**	**1.5**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		5	**5**	**5**	**5**	**5**	3	**5**	**5**	**5**	6.1	6.3	5.9	5.5	7.1	5.4	**1.5**	**1.5**	**0.0**	**0.0**	**0.0**	**0.0**	0.4	**0.0**	**0.0**	**0.0**
125	10	1	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	0.7	1.4	0.9	1.0	1.8	1.2	**0.7**	**0.7**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		3	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	1.8	2.2	1.6	1.5	2.5	1.6	**0.8**	**0.8**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		5	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	2.7	2.8	2.0	1.7	3.3	1.8	**0.9**	**0.9**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	20	1	**5**	**5**	**5**	**5**	4	**5**	**5**	**5**	2.1	2.9	2.6	2.5	3.1	2.7	**1.9**	**1.9**	**0.0**	**0.0**	**0.0**	**0.0**	0.1	**0.0**	**0.0**	**0.0**
		3	1	2	1	1	1	3	4	**5**	4.0	4.3	4.1	4.1	4.7	4.1	**2.4**	**2.4**	1.0	0.9	0.8	0.8	1.2	0.4	0.1	**0.0**
		5	0	0	0	0	0	1	2	**5**	5.8	6.0	5.6	5.5	6.5	5.6	**2.7**	**2.7**	2.5	2.5	2.0	1.6	2.8	1.6	0.5	**0.0**
150	10	1	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	0.5	1.1	0.9	0.9	1.3	1.1	**0.7**	**0.7**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
		3	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	1.6	1.9	1.5	1.4	2.1	1.5	**0.9**	**0.9**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
		5	**5**	**5**	**5**	**5**	3	**5**	**5**	**5**	2.7	2.8	2.4	2.3	3.2	2.6	**1.3**	**1.3**	**0.0**	**0.0**	**0.0**	**0.0**	0.2	**0.0**	**0.0**	**0.0**
	20	1	4	4	4	4	3	4	4	**5**	1.6	2.3	1.9	1.9	2.5	2.2	**1.6**	**1.6**	0.1	0.1	0.2	0.1	0.3	0.1	0.2	**0.0**
		3	0	0	0	0	0	1	1	**3**	3.3	3.7	3.3	3.1	3.9	3.3	**2.1**	**2.1**	1.3	1.3	1.0	0.7	1.6	0.8	0.6	**0.1**
		5	0	0	0	0	0	0	0	**5**	4.7	5.1	4.5	4.3	5.3	4.4	2.6	**2.5**	2.1	2.3	1.8	1.6	2.7	1.6	1.5	**0.0**

Table 6 Nodes and time comparison for KCC instances.

				Av. Nodes Explore								Av. Time (sec)							
n	m	Ω	#	CPLEX	C	CL	BCl	BW	BWL	BG	BGL	CPLEX	C	CL	BCl	BW	BWL	BG	BGL
100	10	1	5	2,187.4	9,493.2	3,407.0	3,808.4	13,278.6	5,055.4	190.8	**169.8**	3.3	2.7	**1.4**	8.9	16.9	9.3	149.4	34.0
		3	5	15,911.0	38,018.4	18,615.8	13,374.4	77,135.4	11,973.4	126.6	**100.2**	9.4	12.1	**6.6**	11.2	51.2	11.2	188.0	52.3
		5	5	56,023.8	73,318.4	34,887.8	11,064.8	197,240.2	24,390.4	181.8	**170.8**	32.3	24.7	13.2	**10.8**	109.3	15.1	431.5	61.1
	20	1	5	12,093.2	84,851.6	36,812.6	37,478.6	137,687.4	56,573.2	2,324.8	**1,116.4**	**19.1**	38.7	19.3	33.9	127.1	45.4	477.1	165.3
		3	5	328,741.6	577,413.8	349,153.2	299,747.8	1,216,771.8	289,571.6	2,545.6	**1,301.0**	375.6	324.8	209.8	179.7	836.9	**176.7**	690.6	246.4
		5	3	796,297.7	1,030,628.0	1,073,455.0	431,804.0	1,648,576.7	655,462.0	1,119.7	**729.0**	970.4	725.8	753.4	**332.2**	1344.0	474.2	1318.0	445.3
125	10	1	5	**2,557.4**	26,109.4	5,254.4	6,706.8	59,136.2	22,296.4	3,862.2	4,874.4	3.3	6.9	**2.4**	15.7	41.8	20.1	111.2	33.5
		3	5	16,810.2	89,336.0	35,721.0	28,278.2	190,739.6	39,481.2	8,113.8	**4,436.2**	**11.8**	23.8	12.8	22.1	88.9	25.3	435.4	71.8
		5	5	157,561.0	301,041.4	253,127.8	87,198.2	1,290,400.0	89,929.6	6,137.0	**4,415.0**	105.5	102.1	84.8	45.9	417.1	**42.8**	863.8	85.1
	20	1	4	243,540.3	1,134,037.8	628,233.8	584,492.3	2,016,914.3	568,772.5	120,848.5	**90,177.3**	357.6	564.6	327.2	332.0	1038.2	288.9	422.5	**237.8**
		3	1	445,816.0	902,386.0	476,674.0	513,874.0	2,206,133.0	1,047,598.0	79,141.0	**29,092.0**	645.0	668.2	**340.7**	431.7	1726.1	663.5	1098.4	631.8
		5	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
150	10	1	5	**1,835.4**	39,164.8	14,878.4	18,218.4	61,288.8	33,738.0	17,672.8	8,190.4	**2.9**	10.0	5.2	23.3	44.7	28.8	101.8	38.3
		3	5	267,752.2	1,156,660.6	1,084,202.4	400,828.6	1,886,360.8	399,942.4	56,666.0	**36,921.0**	183.1	389.2	348.5	146.3	764.2	136.9	358.0	**94.1**
		5	3	394,755.0	877,686.3	423,235.0	337,257.7	2,718,139.7	658,492.3	**87,523.7**	108,409.0	252.5	262.0	157.4	**132.2**	1092.1	218.6	566.7	182.8
	20	1	3	**338,981.0**	2,736,707.7	975,963.7	1,014,534.7	3,681,778.3	2,585,156.0	586,007.0	418,882.0	567.4	1456.4	635.3	632.5	2108.4	1296.5	839.5	**339.8**
		3	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		5	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

D Average Performance Comparison for General Chance Constraints

Tables 7-9 show the number of instances solved, average root gap, and average final gap for each n, m, Ω, and t combination, with $\mathcal{W} = 4000$. Similarly, Tables 10-12 show the average number of nodes in the branch-and-bound search and the average run time for the instances that all techniques solved to optimality. Column # shows the number of instances used to compute the average results.

Table 7 Gap comparison across all techniques for instances with $t = 0.1$

			# Instances Solved to Optimality					Av. Root Gap (%)					Av. Final Gap (%)				
n	m	Ω	CPLEX	BW	BWL	BG	BGL	CPLEX	BW	BWL	BG	BGL	CPLEX	BW	BWL	BG	BGL
75	10	1	**5**	**5**	**5**	**5**	**5**	7.6	10.0	7.1	**3.8**	**3.8**	0.0	0.0	0.0	0.0	0.0
		3	**5**	**5**	**5**	**5**	**5**	47.6	45.6	22.7	16.2	**15.5**	0.0	0.0	0.0	0.0	0.0
		5	**5**	**5**	**5**	**5**	**5**	56.1	48.0	20.2	18.0	**17.1**	0.0	0.0	0.0	0.0	0.0
	20	1	**5**	**5**	**5**	**5**	**5**	54.7	56.5	54.9	**47.0**	47.1	0.0	0.0	0.0	0.0	0.0
		3	**5**	**5**	**5**	**5**	**5**	83.0	76.4	27.9	**26.2**	28.5	0.0	0.0	0.0	0.0	0.0
		5	**5**	**5**	**5**	**5**	**5**	81.3	31.8	9.4	**8.5**	9.4	0.0	0.0	0.0	0.0	0.0
100	10	1	**5**	**5**	**5**	**5**	**5**	6.1	7.5	6.7	**5.4**	**5.4**	0.0	0.0	0.0	0.0	0.0
		3	**5**	**5**	**5**	**5**	**5**	34.3	34.0	27.6	**23.3**	**23.3**	0.0	0.0	0.0	0.0	0.0
		5	**5**	**5**	**5**	**5**	**5**	40.8	39.6	26.8	22.5	**22.4**	0.0	0.0	0.0	0.0	0.0
	20	1	0	0	0	**2**	1	27.1	29.1	27.0	**23.5**	**23.5**	17.7	18.1	13.5	**5.3**	7.5
		3	**5**	**5**	**5**	**5**	**5**	65.3	63.0	48.2	**46.0**	**46.0**	0.0	0.0	0.0	0.0	0.0
		5	**5**	**5**	**5**	**5**	**5**	67.5	60.0	42.6	**39.8**	39.9	0.0	0.0	0.0	0.0	0.0
125	10	1	**4**	**4**	**4**	**4**	**4**	4.4	5.5	4.8	**4.1**	**4.1**	**0.3**	0.4	0.4	**0.3**	0.4
		3	0	0	0	0	0	33.8	33.6	30.6	**29.5**	**29.5**	25.4	23.5	18.1	16.5	**15.0**
		5	0	0	**1**	**1**	**1**	38.0	36.3	32.0	**31.1**	**31.1**	27.9	24.5	13.2	12.5	**9.7**
	20	1	0	0	0	0	0	19.6	20.5	20.0	**18.7**	**18.7**	16.7	17.7	16.7	**15.3**	15.4
		3	0	0	**1**	**1**	**1**	58.0	58.0	53.7	49.7	**49.5**	53.7	51.9	32.4	25.9	**25.2**
		5	0	1	1	**2**	**2**	56.2	55.0	43.3	**40.4**	**40.5**	49.4	21.5	10.2	7.2	**6.1**
Total/Average			59	60	62	**65**	64	43.4	39.5	28.1	**25.2**	25.3	10.6	8.8	5.8	4.6	**4.4**

Table 8 Gap comparison across all techniques for instances with $t = 0.2$

			# Instances Solved to Optimality					Av. Root Gap (%)					Av. Final Gap (%)				
n	m	Ω	CPLEX	BW	BWL	BG	BGL	CPLEX	BW	BWL	BG	BGL	CPLEX	BW	BWL	BG	BGL
75	10	1	**5**	**5**	**5**	**5**	**5**	2.9	4.5	3.6	**2.0**	**2.0**	0.0	0.0	0.0	0.0	0.0
		3	2	4	5	**5**	**5**	10.1	10.7	9.2	**5.4**	**5.4**	1.4	0.3	**0.0**	**0.0**	**0.0**
		5	1	1	3	**5**	**5**	13.1	13.8	11.5	**7.0**	**7.0**	4.5	3.9	1.2	**0.0**	**0.0**
	20	1	2	2	**3**	**3**	**3**	11.6	13.9	11.6	**7.8**	**7.8**	3.2	3.8	2.6	1.4	**0.9**
		3	0	0	0	0	1	31.6	32.4	31.0	**22.1**	**22.1**	25.6	25.6	23.6	14.1	**13.7**
		5	0	0	0	1	1	34.3	34.7	31.7	**22.6**	**22.6**	27.2	28.0	24.8	14.2	**13.9**
100	10	1	**5**	**5**	**5**	**5**	**5**	2.6	3.8	3.1	**2.4**	**2.4**	0.0	0.0	0.0	0.0	0.0
		3	0	1	1	2	**4**	8.3	8.8	8.0	6.3	**6.2**	4.4	4.2	3.3	1.5	**1.0**
		5	0	0	0	0	0	12.4	12.8	12.1	**10.5**	**10.5**	9.0	9.6	8.7	6.9	**6.6**
	20	1	0	0	1	2	**1**	7.8	9.3	8.7	**6.6**	**6.6**	3.7	4.5	3.9	**2.2**	2.4
		3	0	0	0	0	0	26.4	26.7	25.9	**22.6**	**22.6**	23.2	24.2	23.6	19.6	**19.4**
		5	0	0	0	0	0	32.8	32.9	32.0	**28.5**	**28.5**	30.0	30.9	29.5	25.6	**25.5**
125	10	1	**5**	**5**	**5**	**5**	**5**	**1.6**	2.8	2.6	2.4	2.4	0.0	0.0	0.0	0.0	0.0
		3	0	0	0	0	0	7.9	8.1	7.9	**7.6**	**7.6**	5.6	6.0	5.7	5.4	**5.2**
		5	0	0	0	0	0	9.7	9.8	9.5	**8.9**	**8.9**	7.5	8.0	7.9	7.4	**7.4**
	20	1	0	0	0	0	0	**5.6**	6.5	6.3	5.7	5.7	3.7	4.4	4.2	3.7	**3.6**
		3	0	0	0	0	0	23.4	23.5	23.2	21.8	**21.8**	22.0	22.7	22.5	20.7	**20.7**
		5	0	0	0	0	0	26.8	26.8	26.5	**24.5**	**24.5**	25.4	26.1	25.8	23.5	**23.5**
Total/Average			20	23	28	33	**35**	14.9	15.7	14.7	**11.9**	**11.9**	10.9	11.2	10.4	8.1	**8.0**

Table 9 Gap comparison across all techniques for instances with $t = 0.3$

			# Instances Solved to Optimality					Av. Root Gap (%)					Av. Final Gap (%)				
n	m	Ω	CPLEX	BW	BWL	BG	BGL	CPLEX	BW	BWL	BG	BGL	CPLEX	BW	BWL	BG	BGL
75	10	1	**5**	**5**	**5**	**5**	**5**	1.2	2.7	1.8	**0.9**	**0.9**	0.0	0.0	0.0	0.0	0.0
		3	**5**	**5**	**5**	**5**	**5**	3.6	4.4	3.4	**1.8**	**1.8**	0.0	0.0	0.0	0.0	0.0
		5	4	4	**5**	**5**	**5**	4.3	5.0	4.0	**2.1**	**2.1**	0.3	0.4	**0.0**	**0.0**	**0.0**
	20	1	**5**	**5**	**5**	**5**	**5**	3.7	5.5	4.4	**2.3**	2.4	0.0	0.0	0.0	0.0	0.0
		3	0	0	1	**4**	**4**	7.7	9.0	7.6	**4.3**	**4.3**	3.1	3.5	2.2	0.6	**0.2**
		5	0	0	0	3	**4**	8.8	9.8	7.9	**4.2**	4.8	4.8	5.6	3.6	**0.0**	0.2
100	10	1	**5**	**5**	**5**	**5**	**5**	**1.0**	2.1	1.7	1.2	1.2	0.0	0.0	0.0	0.0	0.0
		3	**5**	**5**	**5**	**5**	**5**	2.3	3.1	2.8	**2.1**	**2.1**	0.0	0.0	0.0	0.0	0.0
		5	**5**	**5**	**5**	**5**	**5**	2.9	3.6	3.5	**2.6**	**2.6**	0.0	0.0	0.0	0.0	0.0
	20	1	4	3	4	**5**	**5**	3.3	4.6	4.0	**3.1**	**3.1**	0.2	0.4	0.1	**0.0**	**0.0**
		3	0	0	0	**1**	**1**	6.8	7.5	6.8	**5.4**	**5.4**	4.2	4.8	4.0	2.7	**2.6**
		5	0	0	0	0	0	8.5	9.1	8.6	**6.7**	**6.7**	6.4	6.6	6.1	4.6	**4.4**
125	10	1	**5**	**5**	**5**	**5**	**5**	**0.3**	1.1	1.0	0.9	0.9	0.0	0.0	0.0	0.0	0.0
		3	**5**	**5**	**5**	**5**	**5**	**1.6**	2.2	2.1	1.9	1.9	0.0	0.0	0.0	0.0	0.0
		5	**5**	**5**	**5**	**5**	**5**	**1.8**	2.3	2.1	2.0	2.0	0.0	0.0	0.0	0.0	0.0
	20	1	**5**	4	**5**	**5**	**5**	**1.7**	2.8	2.6	2.4	2.4	**0.0**	0.1	**0.0**	**0.0**	**0.0**
		3	0	0	0	0	0	4.8	5.4	5.2	**4.5**	**4.5**	3.3	3.9	3.5	**3.0**	**3.0**
		5	0	0	0	0	0	7.5	8.0	7.8	**6.9**	**6.9**	6.3	6.7	6.5	5.7	**5.6**
Total/Average			58	56	60	68	**69**	4.0	4.9	4.3	**3.1**	**3.1**	1.6	1.8	1.4	**0.9**	**0.9**

Table 10 Nodes and time comparison for instances solved with $t = 0.1$

				Av. Nodes Explore					Av. Time (sec)				
n	m	Ω	#	CPLEX	BW	BWL	BG	BGL	CPLEX	BW	BWL	BG	BGL
75	10	1	5	12,184.8	16,966.8	5,472.6	544.0	**490.6**	22.8	18.9	10.3	49.6	**17.6**
		3	5	56,780.0	14,815.8	1,003.6	**344.8**	411.6	54.2	16.7	**4.7**	17.1	5.6
		5	5	48,045.0	3,253.4	250.0	278.6	**226.0**	52.5	7.5	**3.6**	10.3	3.7
	20	1	5	202,903.6	215,680.0	146,082.8	27,599.2	**26,056.8**	1,184.5	630.9	526.2	454.0	**309.0**
		3	5	13,220.0	2,036.0	7.0	4.6	**5.6**	58.3	26.1	**11.2**	32.7	11.3
		5	5	11,943.8	23.4	**2.0**	4.4	**2.0**	59.8	15.7	9.2	15.6	**9.0**
100	10	1	5	225,098.4	417,678.4	251,483.8	**79,218.2**	90,030.0	557.1	488.9	367.5	192.1	**152.5**
		3	5	887,680.0	844,533.2	104,978.8	99,247.0	**54,076.8**	1,648.1	1,003.0	185.6	281.8	**115.3**
		5	5	645,651.2	246,868.2	22,797.0	**14,450.2**	18,390.8	1,039.0	372.1	53.5	88.8	**55.1**
	20	1	0	-	-	-	-	-	-	-	-	-	-
		3	5	351,411.2	45,446.2	7,104.4	**5,298.8**	5,562.4	1,733.5	245.2	49.9	108.3	**49.6**
		5	5	270,318.6	10,668.0	2,959.6	**2,201.6**	2,292.8	1,177.4	70.6	30.9	55.4	**30.8**
125	10	1	4	426,022.3	1,342,855.3	935,676.0	**833,522.3**	990,455.3	1,039.1	1,117.8	959.8	**816.5**	910.5
		3	0	-	-	-	-	-	-	-	-	-	-
		5	0	-	-	-	-	-	-	-	-	-	-
	20	1	0	-	-	-	-	-	-	-	-	-	-
		3	0	-	-	-	-	-	-	-	-	-	-
		5	0	-	-	-	-	-	-	-	-	-	-
Average				262,604.9	263,402.1	123,151.5	**88,559.5**	99,000.1	718.9	334.5	184.4	176.9	**139.2**

Table 11 Nodes and time comparison for instances solved with $t = 0.2$

				Av. Nodes Explore					Av. Time (sec)				
n	m	Ω	#	CPLEX	BW	BWL	BG	BGL	CPLEX	BW	BWL	BG	BGL
75	10	1	5	12,121.4	17,446.4	8,475.4	1,150.2	**1,417.8**	20.8	18.4	13.8	42.6	**21.3**
		3	2	467,592.5	533,607.0	181,152.0	34,845.5	**34,824.5**	1,851.3	1,502.2	797.6	146.0	**104.5**
		5	1	465,102.0	501,715.0	260,340.0	**118,609.0**	140,669.0	1,418.2	852.0	489.3	461.8	**451.4**
	20	1	2	188,805.5	359,856.0	126,528.5	**11,088.5**	15,135.0	1,361.3	1,313.8	524.6	317.3	**195.0**
		3	0	-	-	-	-	-	-	-	-	-	-
		5	0	-	-	-	-	-	-	-	-	-	-
100	10	1	5	95,393.2	248,544.4	151,204.4	56,917.2	**30,007.2**	209.1	257.0	168.9	98.0	**63.5**
		3	0	-	-	-	-	-	-	-	-	-	-
		5	0	-	-	-	-	-	-	-	-	-	-
	20	1	0	-	-	-	-	-	-	-	-	-	-
		3	0	-	-	-	-	-	-	-	-	-	-
		5	0	-	-	-	-	-	-	-	-	-	-
125	10	1	5	**49,518.0**	312,285.4	254,152.8	310,771.6	206,355.0	**112.0**	237.6	225.3	244.7	184.6
		3	0	-	-	-	-	-	-	-	-	-	-
		5	0	-	-	-	-	-	-	-	-	-	-
	20	1	0	-	-	-	-	-	-	-	-	-	-
		3	0	-	-	-	-	-	-	-	-	-	-
		5	0	-	-	-	-	-	-	-	-	-	-
Average				213,088.8	328,909.0	163,642.2	88,897.0	**71,401.4**	828.8	696.8	369.9	218.4	**170.1**

Table 12 Nodes and time comparison for instances solved with $t = 0.3$

				Av. Nodes Explore					Av. Time (sec)				
n	m	Ω	#	CPLEX	BW	BWL	BG	BGL	CPLEX	BW	BWL	BG	BGL
75	10	1	5	1,749.0	4,934.2	1,579.0	606.6	**627.8**	**4.7**	11.6	9.5	33.7	20.0
		3	5	105,041.6	316,139.6	88,043.4	3,903.8	**2,959.4**	249.8	422.3	108.3	137.5	**91.0**
		5	4	106,289.8	243,311.0	47,106.5	2,933.0	**1,837.0**	214.8	268.4	81.4	118.6	**57.6**
	20	1	5	46,164.6	84,968.0	54,357.6	6,022.0	**4,415.4**	236.1	349.8	206.0	158.5	**82.3**
		3	0	-	-	-	-	-	-	-	-	-	-
		5	0	-	-	-	-	-	-	-	-	-	-
100	10	1	5	4,668.2	12,624.4	12,106.0	**2,334.0**	3,958.8	**11.0**	20.8	21.7	31.4	24.8
		3	5	98,319.6	234,446.4	245,079.2	43,458.2	**33,129.8**	245.3	373.8	379.2	127.6	**91.3**
		5	5	325,560.8	742,981.0	629,685.2	**137,986.0**	164,530.4	1,081.2	1,147.9	1,441.6	388.6	**432.2**
	20	1	3	73,317.0	103,117.7	112,111.3	48,466.3	**43,590.0**	550.6	307.5	337.8	209.8	**178.7**
		3	0	-	-	-	-	-	-	-	-	-	-
		5	0	-	-	-	-	-	-	-	-	-	-
125	10	1	5	**620.6**	7,116.8	5,396.6	6,666.6	3,225.8	**2.5**	19.2	18.6	23.6	18.8
		3	5	115,889.0	291,387.6	303,281.4	137,235.4	**129,738.6**	301.2	472.1	446.9	237.0	**195.3**
		5	5	**244,464.4**	408,045.8	394,664.0	299,365.8	258,533.4	874.4	645.0	598.8	626.4	**416.9**
	20	1	4	**156,326.3**	674,932.5	420,752.8	385,045.5	253,781.3	**1,279.3**	1,992.8	1,284.0	1,329.8	1,361.9
		3	0	-	-	-	-	-	-	-	-	-	-
		5	0	-	-	-	-	-	-	-	-	-	-
Average				106,534.2	260,333.7	192,846.9	89,501.9	**75,027.3**	420.9	502.6	411.2	285.2	**247.6**