On the Structure of Decision Diagram-Representable Mixed Integer Programs with Application to Unit Commitment

Hosseinali Salemi
Department of Industrial and Manufacturing Systems Engineering, Iowa State University, Ames, IA 50011, hsalemi@iastate.edu

Danial Davarnia
Department of Industrial and Manufacturing Systems Engineering, Iowa State University, Ames, IA 50011, davarnia@iastate.edu

Over the past decade, decision diagrams (DDs) have been used to model and solve integer programming and combinatorial optimization problems. Despite successful performance of DDs in solving various discrete optimization problems, their extension to model mixed integer programs (MIPs) such as those appearing in energy applications has been lacking. More broadly, the question which problem structures admit a DD representation is still open in the DDs community. In this paper, we address this question by introducing a geometric decomposition framework based on rectangular formations that provides both necessary and sufficient conditions for a general MIP to be representable by DDs. As a special case, we show that any bounded mixed integer linear program admits a DD representation through a specialized Benders decomposition technique. The resulting DD encodes both integer and continuous variables, and therefore is amenable to the addition of feasibility and optimality cuts through refinement procedures. As an application for this framework, we develop a novel solution methodology for the unit commitment problem (UCP) in the wholesale electricity market. Computational experiments conducted on a stochastic variant of the UCP show a significant improvement of the solution time for the proposed method when compared to the outcome of modern solvers.

Key words: Decision Diagrams, Benders Decomposition, Mixed Integer Programs, Unit Commitment

1. Introduction

Improved methods for scheduling and dispatching resources are necessary to accommodate the increases in renewable generation, distributed energy resources and storage in the electric grid. At the core of electric grid lies the Unit Commitment Problem (UCP), which determines the thermal unit schedules and generation dispatch in wholesale electricity markets. Unfortunately, the UCP is computationally difficult, making it challenging for available software despite all advances in computing power and technology. For this reason, innovative methodologies that produce optimal solutions to the UCP satisfying realistic yet complicated constraints are needed to manage the increasing amounts of variable renewable generation and distributed energy resources in the electric grid.

Over the past decade, a powerful alternative to traditional solution techniques was developed based on a new paradigm called Decision Diagrams (DDs), in which discrete optimization problems are reconstructed as network models with special structure. The network structure is exploited to speed up the solution process and improve the solution quality, especially for formulations that are poorly handled by conventional solvers. However, since DDs have been designed for modeling discrete problems, they have never been used to solve optimization problems that appear in the electric grid applications, as they naturally contain both discrete and continuous variables. In this paper, we generalize the concept of DDs to mixed integer programs (MIPs), and thereby, establish a novel framework that can be applied to the UCP with the purpose of improving the solution time and quality obtained from existing techniques.

1.1. Background on Decision Diagrams

DDs were introduced in 2006 (Hadžić and Hooker 2006) as a solution method for discrete optimization and combinatorial problems. DDs' principal idea is to model the solutions of the underlying discrete set through a directed acyclic graphical structure with a root and a terminal node. Each path of the graph from the root to the terminal node encodes a solution of the problem where arcs are labeled by values of decision variables on their relative domain. Structurally, DDs resemble branch-and-bound trees and dynamic programming graphs. The key difference, however, is that the size of DDs can be controlled by a width limit factor that mitigates the exponential growth rate intrinsic to branch-and-bound and dynamic programming. This control factor is attributed to a node-merging concept where nodes of the DD are merged to reduce its size at the price of expanding the solution set, thereby yielding a discrete relaxation. By exploiting the concept of relaxed DDs, a specialized branch-and-bound method is designed to successively refine relaxed DDs until proving optimality.

This innovative approach to restructure the graph of the solution set makes DDs a competitive alternative to traditional divide-and-conquer solution techniques such as branch-and-bound and dynamic programming. Various studies in the past decade have been devoted to showing remarkable improvements of solution time and quality achieved by DDs when compared to the outcome of modern solvers. DDs have found their way to a broad array of application areas, from healthcare to supply chain management to finance (Bergman and Cire 2018). Other directions of application include cutting plane theory (Davarnia and van Hoeve 2020), multi-objective optimization (Bergman and Cire 2016), post-optimality analysis (Serra and Hooker 2020), and integrated branch-and-bound (Gonzalez et al. 2020).

While DDs provide a powerful optimization tool, their applicability domain is limited to discrete problems. This limitation is due to a structural requirement that DDs contain a finite number of arcs whose labels represent variable values on their relative domain. Consequently, a successful extension of DDs to model MIPs has been lacking in the literature. Davarnia (2021) took the initiative in this direction by introducing a new concept called *arc-reduction* that enables building relaxed DDs for continuous nonlinear programs. Such a DD is used in a cut-generating oracle to obtain linear valid inequalities for the continuous feasible region of the underlying problem. When embedded inside an outer-approximation scheme, the resulting cutting planes can improve the bounds obtained from classical methods. This cut-generating framework can be viewed as an interface between DDs and traditional cutting plane methods. Such an intermediary role, however, does not utilize the full potential of DDs achieved through specialized branch-and-bound and refinement methods. In the present paper, we develop a framework that *directly* models MIPs through DDs.

1.2. Background on Unit Commitment Problem

The UCP is strongly NP-hard as shown in Bendotti et al. (2019). Over the past three decades, numerous exact and heuristic approaches have been proposed to solve different variants of the UCP and its extensions under both deterministic and stochastic settings, resulting in a rich literature. Examples to solve stochastic/robust UCP include Benders decomposition (Li et al. 2007), two-stage stochastic programming (Blanco and Morales 2017), heuristic scenario reduction methods (Feng and Ryan 2016), two-stage adaptive robust optimization (Bertsimas et al. 2012), and algorithms based on constraint generation (Lorca et al. 2016). We encourage the interested reader to consult a survey by van Ackooij et al. (2018) on solution methods for the UCP under uncertainty. Meanwhile, examples to solve deterministic variants include priority listing methods (Lee and Feng 1992), Tabu search (Rajan et al. 2003), Genetic algorithms (Maifeld and Sheble 1996), dynamic programming (Pang et al. 1981, Frangioni and Gentile 2006), simulated annealing (Mantawy et al. 1998), Lagrangian relaxation (Baldick 1995), fuzzy systems (Saneifard et al. 1997), and MIP formulations. We refer the interested reader to Padhy (2004) for a detailed account on deterministic UCP. Other solution approaches include polyhedral analysis of UCP variants; see Rajan et al. (2005), Queyranne and Wolsey (2017), and Bendotti et al. (2018) for examples.

Garver (1962) is among the pioneers to model the initial variants of the UCP by a MIP that uses three sets of binary variables: (i) on/off variables, (ii) start-up variables, and (iii) shut-down variables. Since then, many efforts have been made to improve MIP formulations either by making them easier to solve through reducing the number of variables, constraints and nonzeros, or by proposing stronger formulations with tighter LP relaxation. Ostrowski et al. (2011) propose a new MIP formulation with three sets of binary variables and show that the computational results are superior to those obtained by using fewer sets of 0-1 variables. Meanwhile, as explained by the authors, one can generate strong valid inequalities when using all three sets of 0-1 variables, which in turn leads to tighter LP relaxations. Subsequently, Morales-España et al. (2013) propose a stronger MIP formulation than those of Carrión and Arroyo (2006) and Ostrowski et al. (2011) with a fewer number of constraints and nonzeros. Interested reader is referred to Wu (2011) and Frangioni et al. (2008) for other examples of MIP models. In addition, a comprehensive analysis and comparison of different MIP formulations are provided by Knueven et al. (2020b).

Due to the substantial cost savings resulted from high quality solutions of the unit commitment, a host of UCP formulations and solution techniques have been proposed to solve real-world instances; see Bertsimas et al. (2012), Atakan et al. (2017), Franz et al. (2020), and Knueven et al. (2020a), for successful implementations.

1.3. Contributions

As discussed earlier, the existing DD-based solution methods are limited to discrete problems only. In this paper, we introduce a framework that generalizes the concept of DDs to solve MIPs directly, which allows for taking advantage of the full arsenal of branch-and-bound and refinement techniques available for DDs. In particular, we propose a geometric concept based on decomposing a MIP solution set into rectangular formations of certain properties, which provides both necessary and sufficient conditions for a general optimization problem to be *representable* by DDs. The significance of this contribution is two-fold: (i) it notably expands the applicability domain of DDs by lifting the integrality barrier; and (ii) it addresses a fundamental open question in the community on which problem structures are amenable to a DD representation. As a consequence, we show that a bounded mixed integer linear program admits a DD representation when reformulated through a specialized Benders decomposition. This result opens a new vein of applications for DDs to solve a broader class of optimization problems. As an evidence for such applications, we develop a novel DD-based solution method to solve the UCP in the energy sector. Being the first DD-based solution technique for the UCP, our results exhibit a great potential in solving this challenging problem class more efficiently.

The remainder of the paper is organized as follows. We develop a rectangular decomposition technique for MIPs and establish its connection to constructing DDs with both integer and continuous variables in Section 2. We transition from the concept of rectangular decomposition to the UCP application through a specialized Benders decomposition technique presented in Section 3. Section 4 is devoted to solving the UCP using the developed DD-based framework. Concluding remarks are given in Section 5.

Due to space restrictions, proofs are omitted from the main part of the paper and can be found in Appendix A. Further, a technical comparison of our framework on the integration of Benders decomposition and DDs with those in the literature is given in Appendix B.

2. Decision Diagrams Representation

In this section, we introduce the concept of rectangular decomposition which bridges a general MIP and DDs. But first, we give a brief overview on the basics of DDs for optimization. A comprehensive review can be found in Bergman et al. (2016).

2.1. Overview on Decision Diagrams

Define $N := \{1, \dots, n\}$. We refer to a DD by $\mathcal{D} = (\mathcal{U}, \mathcal{A}, l(\cdot))$, where $\mathcal{U}$ denotes the set of nodes and $\mathcal{A}$ represents the set of arcs, and function $l : \mathcal{A} \to \mathbb{R}$ indicates the label of arcs in $\mathcal{A}$. The multi-graph induced by $\mathcal{D}$ is composed of n arc layers $\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n$, and $n+1$ node layers $\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_{n+1}$. Node layer $\mathcal{U}_1$ contains a single *root* node r, and node layer $\mathcal{U}_{n+1}$ contains a single *terminal* node t. We define the *width* of a DD as the maximum number of nodes at any node layer, that is $\max\{|\mathcal{U}_i| : i \in \{1, \dots, n+1\}\}$.

DD $\mathcal{D}$ represents a set of points of the form $\boldsymbol{x} = (x_1, \dots, x_n)$ with the following characteristics. The label $l(a)$ of each arc $a \in \mathcal{A}_j$, for $j \in N$, represents the value of x_j. Each arc-sequence (path) from r to t encodes a specific value assignment to $\boldsymbol{x}$ associated with the labels of the arcs on the path. The collection of points encoded by all paths of $\mathcal{D}$ is referred to as the solution set of $\mathcal{D}$, which is denoted by $\text{Sol}(\mathcal{D})$.

Consider a bounded integer set $\mathcal{P} \subseteq \mathbb{Z}^n$. Set $\mathcal{P}$ can be expressed by a DD $\mathcal{D}$ whose collection of all r-t paths encodes the solutions of $\mathcal{P}$, *i.e.,* $\mathcal{P} = \text{Sol}(\mathcal{D})$. Using this definition, we can model discrete problems with DDs. Consider a bounded integer program $z^* = \max\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{P}\}$ where $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathcal{P} \subseteq \mathbb{Z}^n$. To model the above integer program with a DD, we first construct a *weighted* DD, denoted by $[\mathcal{D}|w(\cdot)]$, where (i) $\mathcal{D}$ represents an exact DD encoding solutions of $\mathcal{P}$, and (ii) $w(\cdot) : \mathcal{A} \to \mathbb{R}$ is a weight function associated with arcs of $\mathcal{D}$ so that for each r-t path $\text{P} = (a_1, \dots, a_n)$, its weight

$w(\mathrm{P}) := \sum_{i=1}^{n} w(a_i)$ is equal to $f(\boldsymbol{x}^{\mathrm{P}})$, the objective value of the integral solution corresponding to P. Construction of the weight function is immediate when $f(\boldsymbol{x})$ is separable, i.e., $f(\boldsymbol{x}) = \sum_{i=1}^{n} f_i(x_i)$, as we can simply define $w(a_i) = f_i(l(a_i))$ for $a \in \mathcal{A}_i$ and $i \in N$. The above definition implies that z^* is equal to the weight of a longest path from r to t in the corresponding weighted acyclic graph.

2.2. Rectangular Decomposition

DDs are often described as a union of arc-sequences (paths) from r to t. Since arcs on the path encode certain value assignments to variables, DDs are composed of a finite number of solution points. The key to extend DDs to model continuous variables is to view them as a union of *node-sequences* from r to t. From this perspective, multiple arcs with different label values can be considered simultaneously between two consecutive nodes. We will show later that, in an extreme case, the arcs between two nodes can virtually span an entire continuous interval between two values in the domain of variables. As a result, the node-sequence will encode a hyper-rectangle, as opposed to a single point encoded by the arc-sequence. This analogy lays the foundation for a decomposition technique, which we refer to as *rectangular decomposition*, that restructures the feasible region through a collection of hyper-rectangles with specific properties. This decomposition framework plays a key role in our ability to represent general sets via DDs.

For any $I \subseteq N$, let $\boldsymbol{x}_I$ represent coordinates of $\boldsymbol{x} \in \mathbb{R}^n$ whose index belongs to I. Given a set $P \subseteq \mathbb{R}^n$, $\mathrm{conv}(P)$ describes the convex hull of P, $\mathrm{proj}_{\boldsymbol{x}_I}(P)$ represents the projection of P onto the space of variables $\boldsymbol{x}_I$, and $\dim P$ indicates the dimension of $\mathrm{conv}(P)$. Given a function $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ and a variable subset $I \subseteq N$, we say that $f(\boldsymbol{x})$ *is convex in* $\boldsymbol{x}_I$ if the restriction of $f(\boldsymbol{x})$ to the hyperplane defined by $\{x \in \mathbb{R}^n | x_i = \bar{x}_i, \forall i \in N \setminus I\}$ is convex for any assignment values $\bar{x}_i \in \mathbb{R}$. When we refer to extreme points of a bounded set P, denoted by $\mathcal{X}(P)$, we mean extreme points of $\mathrm{conv}(P)$.

Theorem 1. *Consider a compact set $\mathcal{P} \subseteq \mathbb{R}^n$, and select $I \subseteq N$. Assume that there exists a finite collection of compact sets P_I^j for $j \in J$, where J is an index set, such that* *(i)* $\dim \operatorname{proj}_{x_i}(P_I^j) = 0$, *for* $i \in N \setminus I$ *and* $j \in J$, *i.e., coordinate* x_i *is fixed.*

(ii) $\bigcup_{j\in J} \mathcal{X}(P_I^j) \subseteq \mathcal{P} \subseteq \bigcup_{j\in J} P_I^j$.

(iii) For each $j \in J$, *there exists a finite collection of hyper-rectangles of the form* $R_j^k = \prod_{i=1}^n [l_i^k, u_i^k]$ *for* $k \in K_j$, *where* K_j *is an index set, such that* $\operatorname{conv}(P_I^j) = \operatorname{conv}(\bigcup_{k\in K_j} R_j^k)$.

Then, $\max\{f(\boldsymbol{x})|\boldsymbol{x} \in \mathcal{P}\} = \max\{f(\boldsymbol{x})|\boldsymbol{x} \in \bigcup_{j\in J} \mathcal{X}(P_I^j)\} = \max\{f(\boldsymbol{x})|\boldsymbol{x} \in \bigcup_{j\in J}\bigcup_{k\in K_j} R_j^k\}$ *for any function* $f(\boldsymbol{x})$ *that is convex in* $\boldsymbol{x}_I$. $\square$

We say that set $\mathcal{P}$ *admits* a rectangular decomposition w.r.t. I, if there exists a finite collection of compact sets P_I^j that satisfy conditions (i)–(iii) of Theorem 1. We next illustrate this decomposition concept in an example.

Example 1. Consider a compact mixed integer set

$$\mathcal{P} = \left\{(x_1, x_2, x_3) \in [0,2]^2 \times \{0,1\} \,\middle|\, x_1 + x_2 \le 3;\ x_3 - x_1 \le 0;\ x_1 - x_3 \le 1\right\}.$$

Set $\mathcal{P}$ is shown in Figure 1. Select $I = \{1,2\}$. To use the result of Theorem 1, we first introduce sets $P_I^1 = \{\boldsymbol{x} \in \mathbb{R}^3 \mid 0 \le x_1 \le 1; 0 \le x_2 \le 2; x_3 = 0\}$, and $P_I^2 = \{\boldsymbol{x} \in \mathbb{R}^3 \mid 1 \le x_1 \le 2; 0 \le x_2 \le 2; x_1 + x_2 \le 3; x_3 = 1\}$. It is easy to verify that conditions (i) and (ii) of Theorem 1 hold for these sets as x_3 is fixed in sets P_I^1 and P_I^2, and $\{(0,0,0),(1,0,0),(0,2,0),(1,2,0),(1,1,1),(1,2,1),(2,0,1),(2,1,1)\} \subseteq \mathcal{P} \subseteq P_I^1 \cup P_I^2$. Next, define $J = \{1,2\}, K_1 = \{1\}, K_2 = \{1,2\}$, and hyper-rectangles $R_1^1 = [0,1] \times [0,2] \times \{0\}$ for P_I^1, and $R_2^1 = \{1\} \times [0,2] \times \{1\}$ and $R_2^2 = \{2\} \times [0,1] \times \{1\}$ for P_I^2. These hyper-rectangles satisfy condition (iii) of Theorem 1. As a result, $\max\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{P}\} = \max\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{X}(P_I^1) \cup \mathcal{X}(P_I^2)\} = \max\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in R_1^1 \cup R_2^1 \cup R_2^2\}$ for any function $f(\boldsymbol{x})$ that is convex in (x_1, x_2).

In view of Theorem 1, two levels of decomposition are performed. First, $\mathcal{P}$ is decomposed into sets P_I^j through fixing variables not in I. Second, each set P_I^j is decomposed into hyper-rectangles through a convex hull description. The second level can be viewed as a generalization of the standard extreme point decomposition theorem for maximizing convex functions (Rockafeller 1970), as the hyper-rectangles can be chosen to be the extreme points. We note, however, that the addition of the first level is necessary to account for the non-convex coordinates of the objective function.

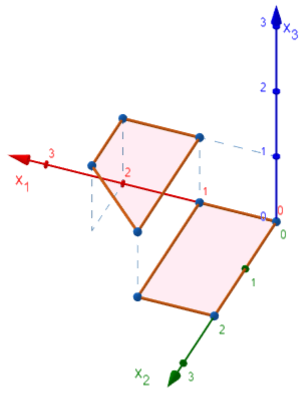


Figure 1 Set $\mathcal{P}$ of Example 1

For instance, consider $\mathcal{P} = \{(x_1, x_2) \in [-1,1] \times \{-1,0,1\} \mid |x_1| \leq |x_2|\}$, and select $I = \{1\}$. Clearly $\mathcal{X}(\mathcal{P}) = \{(-1,-1),(-1,1),(1,-1),(1,1)\}$. It follows that the extreme point decomposition alone does not provide the desired equivalence as $\max\{-x_2^2 \mid \boldsymbol{x} \in \mathcal{P}\} \neq \max\{-x_2^2 \mid \boldsymbol{x} \in \mathcal{X}(\mathcal{P})\}$. Further, we will discuss later that the rectangular volume of the decomposition is key to build efficient DDs as compared to the case for extreme points. Next, we present a partial converse of Theorem 1 as a sufficient condition for a set to admit rectangular decomposition.

Proposition 1. *Consider a compact set $\mathcal{P} \subseteq \mathbb{R}^n$, and select $I \subseteq N$. Assume that there exists a finite set $\mathcal{Q} \subseteq \mathbb{R}^n$ such that* $\max\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{P}\} = \max\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{Q}\}$ *for every function $f(\boldsymbol{x})$ that is convex in $\boldsymbol{x}_I$. Then, $\mathcal{P}$ admits a rectangular decomposition w.r.t. I.* □

The above rectangular decomposition method has two important consequences in DDs context that will be demonstrated in the next two subsections.

2.3. Equivalence Class

In this section, we establish an equivalence relation between DDs that produce the same optimal value for a given objective function. This result sets the stage for modeling continuous variables via DDs.

Lemma 1. *Consider a compact set $\mathcal{P} \subseteq \mathbb{R}^n$, and select $I \subseteq N$. Assume that there exists a finite collection of hyper-rectangles of the form $R_I^j = \prod_{k=1}^n [l_k^j, u_k^j]$ for $j \in J$, where J is an index set, such that* *(i)* $l_i^j = u_i^j$ *for all* $i \in N \setminus I$ *and* $j \in J$,

(ii) $\bigcup_{j \in J} \mathcal{X}(R_I^j) \subseteq \mathcal{P} \subseteq \bigcup_{j \in J} R_I^j$.

Then, $\max\{f(\boldsymbol{x}) | \boldsymbol{x} \in \mathcal{P}\} = \max\{f(\boldsymbol{x}) | \boldsymbol{x} \in \bigcup_{j \in J} \mathcal{X}(R_I^j)\} = \max\{f(\boldsymbol{x}) | \boldsymbol{x} \in \bigcup_{j \in J} R_I^j\}$ *for any function* $f(\boldsymbol{x})$ *that is convex in* $\boldsymbol{x}_I$. $\square$

The difference between the result of Lemma 1 and that of Theorem 1 stems from the levels of decomposition involved in the process. In Lemma 1, a set is directly decomposed into rectangular components, whereas in Theorem 1, a preliminary level of decomposition is performed. This additional level, through definition of sets P_I^j, makes it possible to decompose sets that cannot be directly represented through union of hyper-rectangles. For instance, it is easy to verify that set $\mathcal{P}$ of Example 1 cannot be directly decomposed into hyper-rectangles prescribed in Lemma 1, whereas it admits rectangular decomposition through the use of Theorem 1.

In view of Lemma 1, we say that the hyper-rectangles R_I^j that satisfy the lemma assumptions form an *equivalence class* w.r.t. I, as they share the optimal value of the objective function over any set enveloped between R_I^j and their extreme points. We next present a similar equivalence class for DD formulations. For notational convenience, we use label value l_a for an arc $a \in \mathcal{A}$ as a shorthand for $l(a)$. The DDs considered here are not limited by the unique arc-label rule, i.e., they can contain several arcs with equal label values at a layer. For uniformity of arguments, we refer to a *virtual* DD as one that contains, between two nodes, a collection of arcs with label values that span a closed continuous interval; see Davarnia (2021) for an example.

Consider a DD $\mathcal{D} = (\mathcal{U}, \mathcal{A}, l(\cdot))$. For any pair (u, v) of nodes of $\mathcal{D}$, define $A(u, v)$ to be the set of all arcs of $\mathcal{D}$ directed from u to v. Further, define $l_{(u,v)}^{\max}$ (resp. $l_{(u,v)}^{\min}$) to be the maximum (resp. minimum) label of the arcs in a non-empty set $A(u, v)$.

PROPOSITION 2. *Consider a DD* $\mathcal{D} = (\mathcal{U}, \mathcal{A}, l(\cdot))$, *and select* $I \subseteq N$.

(i) Let $\bar{\mathcal{D}}$ *be a DD constructed from* $\mathcal{D}$ *with a difference that, for any node pair* $(u, v) \in \mathcal{U}_i \times \mathcal{U}_{i+1}$ *and any* $i \in I$, *only the arcs with label values* $l_{(u,v)}^{min}$ *and* $l_{(u,v)}^{max}$ *are maintained and the rest are removed.*

(ii) Let $\hat{\mathcal{D}}$ be a virtual DD constructed from $\mathcal{D}$ with a difference that, for any node pair $(u,v) \in \mathcal{U}_i \times \mathcal{U}_{i+1}$ and any $i \in I$, the collection of arcs with label values spanning the interval $[l^{min}_{(u,v)}, l^{max}_{(u,v)}]$ is added between u and v.

Then, $\max\{f(\boldsymbol{x}) | \boldsymbol{x} \in \mathrm{Sol}(\mathcal{D})\} = \max\{f(\boldsymbol{x}) | \boldsymbol{x} \in \mathrm{Sol}(\bar{\mathcal{D}})\} = \max\{f(\boldsymbol{x}) | \boldsymbol{x} \in \mathrm{Sol}(\hat{\mathcal{D}})\}$ *for any function $f(\boldsymbol{x})$ that is convex in $\boldsymbol{x}_I$.* $\square$

Proposition 2 defines an equivalence class for DDs. In words, when there are multiple arcs between two consecutive nodes of a DD, as long as the objective function is convex in the variable corresponding to that arc layer, we can add or remove the middle arcs, and yet preserve the optimal value. This equivalence property enables us to model continuous intervals by translating virtual DDs into regular DDs. This property can also be useful from a computational perspective as the size of the DDs can be reduced through removing unnecessary arcs or through adding arcs that provide more efficient merging possibilities.

2.4. DD-Representable Sets

An important and fundamental question in DDs community concerns the characteristics of optimization problems that can be represented through a DD formulation.

Definition 1. We say that a compact set $\mathcal{P} \subseteq \mathbb{R}^n$ is *DD-representable* w.r.t. an index set I, if there exists a DD $\mathcal{D}$ such that $\max\{f(\boldsymbol{x}) | \boldsymbol{x} \in \mathcal{P}\} = \max\{f(\boldsymbol{x}) | \boldsymbol{x} \in \mathrm{Sol}(\mathcal{D})\}$ for every function $f(\boldsymbol{x})$ that is convex in $\boldsymbol{x}_I$.

In the above definition, the requirement that a set and its associated DD must have the same optimal value for *every* objective function—under appropriate convexity assumptions—is critical for DD-representability. There are several DD-based procedures, such as Lagrangian relaxation methods (Bergman et al. 2015), that use DDs with varying objective functions in different stages of the procedure, showing the necessity of properties that guarantee the conservation of optimal values throughout stages.

The question of whether a given set is DD-representable is straightforward when the set is bounded and discrete. In this case, every point of the set can be encoded by a unique r-t path of a

DD, leading to a finite width limit. This argument, however, does not hold when the set contains continuous variables, as all solutions cannot be fully encoded by finitely many paths in a DD. In light of the rectangular decomposition technique introduced above, we next give necessary and sufficient conditions for a mixed integer set to be DD-representable.

COROLLARY 1. *Consider a compact set $\mathcal{P} \subseteq \mathbb{R}^n$, and select $I \subseteq N$. The set $\mathcal{P}$ is DD-representable w.r.t. I if and only if it admits a rectangular decomposition w.r.t. I.* $\square$

As mentioned above, the conditions of Corollary 1 are immediately satisfied for bounded discrete sets. For mixed integer sets, there is an important class that also satisfies the conditions of Corollary 1 as given next.

COROLLARY 2. *Let $\mathcal{Q} \subseteq \mathbb{R}^{n+1}$ be a compact set and define the mixed integer set $\mathcal{P} = \{(\boldsymbol{x}; y) \in \mathcal{Q} \,|\, \boldsymbol{x} \in \mathbb{Z}^n\}$. Then, $\mathcal{P}$ is DD-representable w.r.t. $I = \{n+1\}$.* $\square$

The class of mixed integer sets that contain a single continuous variable plays an important role in optimization as it can be viewed as the bridge between a pure discrete case and a general mixed integer case; see e.g. the derivation of mixed integer rounding inequalities that is built upon this analogy (Nemhauser and Wolsey 1990, Conforti et al. 2014b). The ability to model this class by DDs, therefore, opens pathways to a wide range of new application domains. The approach is to represent the continuous components of a general mixed integer set by a new variable and develop its corresponding DD, while accounting for the relation between the continuous components and the substitute variable separately. The Benders decomposition technique provides an ideal framework for such an approach. We will establish this framework in Section 3 and apply it to the unit commitment problem in Section 4.

In view of Corollary 2, we remark that when a set involves multiple continuous variables, the existence of a rectangular decomposition is not guaranteed. For example, the unit disk in $\mathbb{R}^2$ described by $x_1^2 + x_2^2 \leq 1$ does not admit a rectangular decomposition; hence it is not DD-representable.

We conclude this section by presenting a key role of the rectangular decomposition technique in determining the DD size, as the main factor in designing efficient DD-based solution methods.

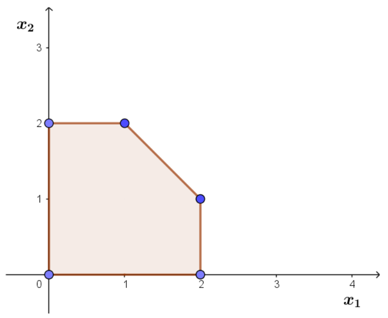
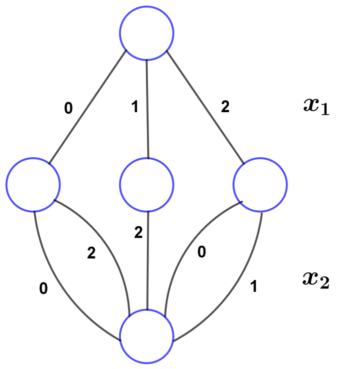


Figure 2 Set $\text{proj}_{(x_1,x_2)}(\mathcal{P})$ of Example 2

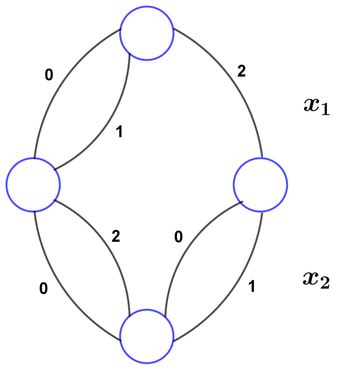
This implication follows from the fact that the rectangular decomposition of a set is not unique, and it can be achieved in different ways. For each decomposition, the resulting hyper-rectangles form an equivalence class for DDs representing them. This variety can be exploited to build DDs with smaller sizes that are computationally more efficient, as illustrated in the next example.

EXAMPLE 2. Consider the set defined by $\text{proj}_{(x_1,x_2)}(\mathcal{P})$ of Example 1; see Figure 2. The goal is to build a DD $\mathcal{D}$ such that $\max\{f(\boldsymbol{x})|\boldsymbol{x} \in \mathcal{P}\} = \max\{f(\boldsymbol{x})|\boldsymbol{x} \in \text{Sol}(\mathcal{D})\}$ for every convex function $f(\boldsymbol{x})$. Define $P^1_I = \mathcal{P}$ with $I = \{1,2\}$. An immediate rectangular decomposition is obtained through rectangles R^j_1 that represent extreme points $\{(0,0),(0,2),(1,2),(2,0),(2,1)\}$ of $\mathcal{P}$. The DD $\mathcal{D}_1$ that models these extreme points forms an equivalence class. The minimum width of $\mathcal{D}_1$ (with the natural ordering of variables in layers) is three, as shown in Figure 3a. As an alternative decomposition, we may define hyper-rectangles $R^1_1 = [0,1] \times [0,2]$ and $R^2_1 = \{2\} \times [0,1]$. The reduced DD $\mathcal{D}_2$ representing this equivalence class has width two, Figure 3b. This shows the practical advantage of DDs as a member of equivalence classes obtained from different rectangular decompositions.

It follows from Example 2 that to reduce the size of the DD representation of a given set, one may seek the solution to the combinatorial problem that finds the minimum number of hyper-rectangles that achieve a rectangular decomposition of the set. This observation implies that a decomposition that contains higher-dimensional hyper-rectangles is generally preferred to one that is simply composed of extreme points (i.e., zero-dimensional rectangles) of the set, as the former can be modeled via fewer node-sequences in a DD.



(a) $\mathcal{D}_1$ with width 3



(b) $\mathcal{D}_2$ with width 2

Figure 3 The impact of equivalence class on DDs width. Numbers next to arcs represent the labels.

3. Benders Decomposition

As an application of the rectangular decomposition technique developed in Section 2.2, we next present a specialized Benders decomposition (BD) framework that uses DDs to solve MIPs.

Consider a bounded MIP $\mathcal{H} = \max\{\boldsymbol{ax} + \boldsymbol{by} \,|\, A\boldsymbol{x} + B\boldsymbol{y} \leq c,\, \boldsymbol{x} \in \mathbb{Z}^n\}$ where $\boldsymbol{a}$ and $\boldsymbol{b}$ are row vectors. We use BD to define the master problem $\mathcal{M} = \max\{\boldsymbol{ax} + z \,|\, (\boldsymbol{x}, z) \in \mathcal{P} \times [l_0, u_0]\}$ where $\mathcal{P} \subseteq \mathbb{Z}^n$ is the projection of the feasible region of $\mathcal{H}$ onto $\boldsymbol{x}$-space, and the bounds on z are induced from the boundedness of $\mathcal{H}$. Subproblems are defined as $\mathcal{S}(\bar{\boldsymbol{x}}) = \max\{\boldsymbol{by} \,|\, B\boldsymbol{y} \leq c - A\bar{\boldsymbol{x}}\}$ for a given $\bar{\boldsymbol{x}}$. At each iteration of the algorithm, the output of the subproblems is either a feasibility cut of the form $\boldsymbol{\alpha x} \leq \alpha_0$ or an optimality cut of the form $z + \boldsymbol{\alpha x} \leq \alpha_0$, where $\boldsymbol{\alpha}$ is a row vector of matching dimension. These cuts are added to the master problem and the problem is resolved.

It follows from Corollary 2 that $\mathcal{M}$ admits a DD representation, forming an equivalence class. Such a DD contains n layers corresponding to integer variables $\boldsymbol{x}$ and one (last) layer corresponding to the continuous variable z with arc labels representing a lower and an upper bound on this variable. Using this DD, we can find its longest r-t path, solve the subproblems for the solution encoding that path, and *refine* the DD with respect to the generated optimality and feasibility cuts. The refinement operation separates the solutions violating the cuts from the DD solution set; see Bergman et al. (2016) for a detailed account on refinement techniques in DD. The main difficulty, however, is that the size of a DD that gives an exact representation of $\mathcal{M}$ grows exponentially with the problem size. It is then imperative to employ the notion of *restricted* and *relaxed* DDs in our algorithm to increase efficiency. The idea is to design DDs with a limited width that represent a restriction (resp. relaxation) of the underlying problem, and thereby providing a primal (resp. dual) bound. The construction of a restricted DD is straightforward, as it can be achieved by selecting any subset of the r-t paths of the exact variant that satisfies the width limit. The construction of a relaxed DD, however, requires a careful manipulation of the DD structure through a so-called *node-merging* operation in such a way that the resulting DD with a smaller width contains all feasible r-t paths of the exact variant, with a possible addition of some infeasible paths. Such a construction exploits the combinatorial structure of $\mathcal{M}$, and hence is problem-specific. We give an instance for designing relaxed DDs for the unit commitment application in Section 4.

Algorithm 1 presents the full DD-based framework to solve $\mathcal{H}$ through BD. In this approach, referred to as DD-BD, we use the following definitions. Let C be the set of optimality and feasibility cuts, and define $\mathcal{F}^C$ to be the feasible region described by constraints of C. Consider $\hat{\boldsymbol{x}} = (\hat{x}_1, \dots, \hat{x}_{|\hat{x}|})$ to be a partial assignment of size $|\hat{x}|$ to variables $x_1, \dots, x_{|\hat{x}|}$. Define $\mathcal{M}^C(\hat{\boldsymbol{x}}) = \max\{\boldsymbol{a}\boldsymbol{x} + z \,|\, (\boldsymbol{x}, z) \in \mathcal{P} \times [l_0, u_0] \cap \mathcal{F}^C, \, x_i = \hat{x}_i, \forall i = 1, \dots, |\hat{x}|\}$ to be the restriction of the master problem $\mathcal{M}$ after adding the cuts in C and fixing the partial assignment $\hat{\boldsymbol{x}}$. We denote the case with an empty partial assignment and empty constraint set as input by $\mathcal{M}^{\emptyset}(\emptyset) = \mathcal{M}$. We assume that oracles to build relaxed and restricted DDs for the master problem $\mathcal{M}$ are available. Using these oracles, one can simply construct relaxed and restricted DDs for $\mathcal{M}^C(\hat{\boldsymbol{x}})$ by fixing the path associated with the partial assignment $\hat{\boldsymbol{x}}$ and refining the DD with respect to the cuts in C. Given a DD $\mathcal{D}$ with n layers, we denote by $\texttt{EXACT}(\mathcal{D})$ the last exact node layer of $\mathcal{D}$, i.e., the last node layer that contains no merged nodes; see Bergman et al. (2016) for properties of the exact cut set operator.

Next, we show the finiteness and correctness of Algorithm 1, followed by some implementation remarks.

Algorithm 1: DD-BD

```
Data: $\mathcal{H}$, $\mathcal{M}$, $\mathcal{S}(\bar{\boldsymbol{x}})$, and oracles to build relaxed and restricted DDs for $\mathcal{M}$
Result: An optimal solution $(\boldsymbol{x}^*, z^*)$ and optimal value $w^*$ of $\mathcal{H}$
1  initialize set of partial assignments $\hat{\mathcal{X}} := \{\emptyset\}$, set of cuts $C := \emptyset$, and lower bound $w^* := -\infty$
2  while $\hat{\mathcal{X}} \neq \emptyset$ do
3      pick $\hat{\boldsymbol{x}} \in \hat{\mathcal{X}}$ and update $\hat{\mathcal{X}} \leftarrow \hat{\mathcal{X}} \setminus \{\hat{\boldsymbol{x}}\}$
4      create a restricted DD $\underline{\mathcal{D}}$ associated with $\mathcal{M}^C(\hat{\boldsymbol{x}})$
5      if $\underline{\mathcal{D}} \neq \emptyset$ then
6          repeat
7              find a longest $r$-$t$ path in $\underline{\mathcal{D}}$ with encoding point $(\underline{\boldsymbol{x}}, \underline{z})$ and length $\underline{w}$
8              solve $\mathcal{S}(\underline{\boldsymbol{x}})$ to obtain feasibility/optimality cuts $\underline{C}$ and optimal value $\underline{\rho}$
9              update $C \leftarrow C \cup \underline{C}$, and refine $\underline{\mathcal{D}}$ w.r.t $\underline{C}$
10         until $\underline{z} = \underline{\rho}$;
11         if $\underline{w} > w^*$ then
12             update $w^* \leftarrow \underline{w}$ and $(\boldsymbol{x}^*, z^*) \leftarrow (\underline{\boldsymbol{x}}, \underline{z})$
13     else
14         Go to line 2
15     create a relaxed DD $\overline{\mathcal{D}}$ associated with $\mathcal{M}^C(\hat{\boldsymbol{x}})$
16     find a longest $r$-$t$ path in $\overline{\mathcal{D}}$ with encoding point $(\overline{\boldsymbol{x}}, \overline{z})$ and length $\overline{w}$
17     if $\overline{w} > w^*$ then
18         repeat
19             solve $\mathcal{S}(\overline{\boldsymbol{x}})$ to obtain feasibility/optimality cuts $\overline{C}$ and optimal value $\overline{\rho}$
20             update $C \leftarrow C \cup \overline{C}$, refine $\overline{\mathcal{D}}$ w.r.t $\overline{C}$, and update its longest $r$-$t$ path solution $(\overline{\boldsymbol{x}}, \overline{z})$ and length $\overline{w}$
21         until $\overline{z} = \overline{\rho}$;
22         For each node $u \in \texttt{EXACT}(\overline{\mathcal{D}})$, add the partial assignment encoding a longest $r$-$u$ path of $\overline{\mathcal{D}}$ to $\hat{\mathcal{X}}$
23     else
24         Go to line 2
25 Return $(\boldsymbol{x}^*, z^*)$ and $w^*$
```

Theorem 2. *Algorithm 1 returns an optimal solution and optimal value of $\mathcal{H}$ in a finite number of iterations.* $\square$

Remark 1. For a selected partial assignment $\hat{\boldsymbol{x}} \in \hat{\mathcal{X}}$ in line 3 of Algorithm 1, if the restricted DD $\underline{\mathcal{D}}$ constructed in line 4 models an exact representation of $\mathcal{M}^C(\hat{\boldsymbol{x}})$, then the construction of a relaxed DD in lines 15–24 can be entirely skipped for that loop. This follows from the fact that the lower bound at this partial assignment cannot be improved any further by branching down through exact cut sets.

Remark 2. The addition of cuts generated from the subproblems to C to be carried over to next iterations is not necessary for the correctness of Algorithm 1 as shown in the proof of Theorem 2. The advantage of this addition is to avoid generating the same cuts at different iterations for next restricted or relaxed DDs. In fact, even invoking subproblems for relaxed DDs (in lines 18–21) can be skipped without invalidating correctness of the algorithm. The addition of this subroutine helps improve the dual bound obtained from the relaxed DDs, and thereby, trigger the pruning condition of line 17 faster.

The next example illustrates steps of Algorithm 1.

Example 3. Consider the following MIP with two integer and two continuous variables.

$$\max_{\boldsymbol{x}\in\{0,1\}^2;\ \boldsymbol{y}\in\mathbb{R}^2_+} \left\{ x_1 + x_2 + 2y_1 + y_2 \,\middle|\, \begin{smallmatrix} x_1+x_2\geq 1 \\ y_1+y_2\geq x_1+x_2 \\ 0.3y_1+0.7y_2\leq 0.1x_1+0.3 \end{smallmatrix} \right\}.$$

The optimal solution of this problem is $(x_1^*, x_2^*, y_1^*, y_2^*) = (1, 0, 1.33, 0)$ with the optimal value 3.66. We intend to solve this problem using the DD-BD approach of Algorithm 1. The master problem $\mathcal{M}$ is formulated as $\max_{\boldsymbol{x}\in\{0,1\}^2} \{x_1 + x_2 + z \,|\, x_1 + x_2 \geq 1\}$, where z represents the objective value of the subproblem $\mathcal{S}(\bar{x}_1, \bar{x}_2)$ described by

$$\max_{\boldsymbol{y}\in\mathbb{R}^2_+} \left\{ 2y_1 + y_2 \,\middle|\, \begin{smallmatrix} y_1+y_2\geq \bar{x}_1+\bar{x}_2 \\ 0.3y_1+0.7y_2\leq 0.1\bar{x}_1+0.3 \end{smallmatrix} \right\}.$$

Defining the dual vector $\boldsymbol{\pi} \in \mathbb{R}^2_+$, we obtain the dual of the above subproblem as follows.

$$\min_{\boldsymbol{\pi}\geq 0} \left\{ -(\bar{x}_1 + \bar{x}_2)\pi_1 + (0.1\bar{x}_1 + 0.3)\pi_2 \,\middle|\, \begin{smallmatrix} -\pi_1+0.3\pi_2\geq 2 \\ -\pi_1+0.7\pi_2\geq 1 \end{smallmatrix} \right\}.$$

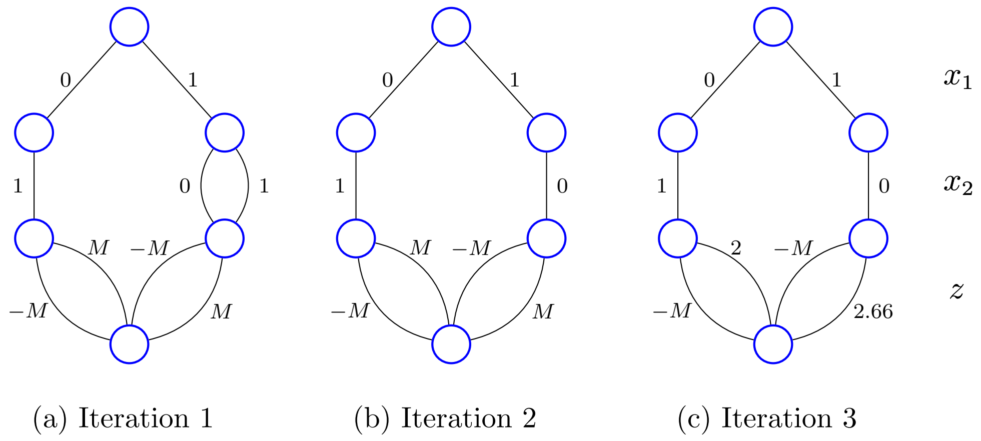


Figure 4 Different iterations of solving the master problem of Example 3.

The feasibility cuts are of the form $\tilde{\pi}_1(x_1+x_2)-\tilde{\pi}_2(0.1x_1+0.3)\leq 0$ where $\tilde{\pi}$ is a recession ray of the dual subproblem. Similarly, the optimality cuts are of the form $z+\hat{\pi}_1(x_1+x_2)-\hat{\pi}_2(0.1x_1+0.3)\leq 0$ where $\hat{\pi}$ is a feasible solution of the dual problem. We create the exact DD $\underline{\mathcal{D}}$ of Figure 4a to represent $\mathcal{M}$ where $-M$ and M are assumed to be some valid bounds on variable z. The longest path of $\underline{\mathcal{D}}$, at this iteration, is associated with the solution $(\underline{x_1},\underline{x_2},\underline{z})=(1,1,M)$ with length $\underline{w}=2+M$. Solving $S(\underline{\boldsymbol{x}})$ gives a feasibility cut $0.66x_1+x_2\leq 1$. We update the set of cuts C and refine $\underline{\mathcal{D}}$ w.r.t to the feasibility cut. The output is the new DD depicted in Figure 4b. At this iteration, a longest path corresponding to the solution $(\underline{x_1},\underline{x_2},\underline{z})=(0,1,M)$ gives the optimal solution $\hat{\boldsymbol{\pi}}=(0,6.66)$ and the optimal value $\underline{\rho}=2$ for the dual subproblem, generating the optimality cut $z\leq 0.66x_1+2$ through solving $S(\underline{\boldsymbol{x}})$. Refining $\underline{\mathcal{D}}$ for the last time w.r.t this optimality cut, the DD of Figure 4c is obtained. It follows that the longest path has length $\underline{w}$ equal to 3.66 and is achieved by $(\underline{x_1},\underline{x_2},\underline{z})=(1,0,2.66)$. At this point, we update $w^*=\underline{w}=3.66$ and $(\boldsymbol{x}^*,z^*)=(\underline{\boldsymbol{x}},\underline{z})=(1,0,2.66)$. Since $\underline{\mathcal{D}}$ models an exact representation of $\mathcal{M}^C(\boldsymbol{x})$, it follows from Remark 1 that we can skip creating relaxed DDs, which leads to terminating the algorithm with the optimal solution $(\boldsymbol{x}^*,z^*)=(1,0,2.66)$ and the optimal value $w^*=3.66$.

While DDs have been used in conjunction with BD for mixed integer models in the literature as in van der Linden (2017), their layer construction has been limited to integer variables only, which has led to major computational shortcomings such as not admitting reduced forms—a critical feature for building efficient DDs in practice. Such limitations are not present in our approach, as it directly incorporates continuous variables within DD layers. We refer the reader to Appendix B for a detailed comparison.

4. Application to the Unit Commitment Problem

In this section, we provide an evidence of practicality for the DD-BD framework by applying it to the UCP—a well-known problem in electrid grid applications. We first present a common MIP formulation for the UCP. To apply our framework, we consider the standard BD formulation of the problem, and through a projective transformation, make it amenable to DD representation. We close the section by comparing the computational results of the DD-BD approach with that of the standard BD formulation.

4.1. Standard Formulations

In the UCP variant we consider in this paper, we seek to find a power generation schedule that minimizes the total operational cost subject to typical UCP constraints such as minimum up/down times, generation capacity, demand satisfaction, ramping requirements, and spinning reserve. The typical MIP formulation of this problem is given in (1a)–(1l); see Rajan et al. (2005), Ostrowski et al. (2011), Tuffaha and Gravdahl (2013), Bendotti et al. (2018). In this formulation, binary variables x^i_j represent whether or not generating unit $i \in N = \{1, \dots, n\}$ is up at time $j \in \mathcal{T} = \{1, \dots, T\}$. Similarly, binary variables y^i_j (resp. $\bar{y}^i_j$) indicate whether or not unit i starts up (resp. shuts down) at time j. Further, continuous variables p^i_j and $\bar{p}^i_j$ denote the production and maximum power available of unit i at time j.

The objective function (1a) of the UCP minimizes the total fixed operating costs c^i_f, production costs c^i_g, and logarithmic start-up costs q^i_j for all units over the planning time horizon. The fixed operating cost c^i_f is paid whenever unit i is up and working. The production cost c^i_g of unit i models the variable cost per unit of generated power. The start-up cost q^i_j of generator i at time j is a logarithmic function of the inactive duration of the generator to model the fact that: the colder a generator gets, the greater cost is incurred to start it again. This function is commonly linearized by discretizing the time horizon and evaluating the start-up costs K^i_k of unit i after it has been inactive for k consecutive time periods (Ostrowski et al. 2011, Tuffaha and Gravdahl 2013). This linearization step is modeled in constraint (1b). Constraint (1c) represents the logical relation between the commitment variables x^i_j and variables y^i_j and $\bar{y}^i_j$. constraints (1d) and (1e) model the requirement that the schedule of each generator i must satisfy a minimum down-time $\ell^i \geq 1$ and a minimum up-time $L^i \geq 1$, i.e., if a generator i shuts down (resp. starts up) at time j, then it must be down (resp. up) for at least ℓ^i (resp. L^i) time periods. In addition, each generator i has a start-up SU^i, ramp-up RU^i, shut-down SD^i, and ramp-down RD^i rates. These parameters bound the changes in production rate of generators in consecutive time periods. For instance, if unit i is down at time j, then its production is bounded above by SU^i at time $j+1$. Similarly, if unit i is working at time j with a generation output strictly greater than SD^i, then it cannot be offline at time $j+1$. These requirements are represented in constraints (1f) and (1g) as given in Tuffaha and Gravdahl (2013). It is commonly assumed that $SU^i \leq RU^i$ and $SD^i \leq RD^i$ for all $i \in N$. Further, each generator i has a minimum m^i and a maximum M^i production capacity that are captured in constraints (1h). Finally, constraints (1i) and (1j) guarantee that the total demand D_j and the spinning reserve requirement R_j are satisfied at each time period j.

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{T} c^i_f x^i_j + c^i_g p^i_j + q^i_j \tag{1a}$$

$$\text{s.t.} \quad q^i_j \geq K^i_k \left(x^i_j - \sum_{h=1}^{k} x^i_{j-h} \right) \qquad \forall k \in \{1,\dots,j-1\}, \quad \forall j \in \mathcal{T}, \quad \forall i \in N \tag{1b}$$

$$y^i_j - \bar{y}^i_j = x^i_j - x^i_{j-1} \qquad \forall j \in \{1,\dots,T\}, \quad \forall i \in N \tag{1c}$$

$$\sum_{j'=j-L^i+1}^{j} y^i_{j'} \leq x^i_j \qquad \forall j \in \{L^i,\dots,T\}, \quad \forall i \in N \tag{1d}$$

$$\sum_{j'=j-\ell^i+1}^{j} \bar{y}^i_{j'} \leq 1 - x^i_j \qquad \forall j \in \{\ell^i,\dots,T\}, \quad \forall i \in N \tag{1e}$$

$$p^i_j - p^i_{j-1} \leq RU^i x^i_{j-1} + SU^i y^i_j \qquad \forall j \in \mathcal{T}, \quad \forall i \in N \tag{1f}$$

$$p^i_{j-1} - p^i_j \leq RD^i x^i_j + SD^i \bar{y}^i_j \qquad \forall j \in \mathcal{T}, \quad \forall i \in N \tag{1g}$$

$$m^i x^i_j \leq p^i_j \leq \bar{p}^i_j \leq M^i x^i_j \qquad \forall j \in \mathcal{T}, \quad \forall i \in N \tag{1h}$$

$$\sum_{i=1}^{n} p^i_j \geq D_j \qquad \forall j \in \mathcal{T} \tag{1i}$$

$$\sum_{i=1}^{n} \bar{p}^i_j \geq D_j + R_j \qquad \forall j \in \mathcal{T} \tag{1j}$$

$$x^i_j, y^i_j, \bar{y}^i_j \in \{0,1\} \qquad \forall j \in \mathcal{T}, \quad \forall i \in N \tag{1k}$$

$$q^i_j, p^i_j, \bar{p}^i_j \geq 0 \qquad \forall j \in \mathcal{T}, \quad \forall i \in N. \tag{1l}$$

The above MIP formulation is used as the core model for the two-stage stochastic UCP, where the first stage decides the commitment status of units, and the second stage determines the generation schedule to satisfy uncertain demand represented by a number of possible scenarios. Due to the *L-shaped* structure of this stochastic UCP, a BD method is commonly used to solve the problem; see (Zheng et al. 2013, 2014) for examples of such approach. Other reasons advocating use of BD include slow convergence of the full MIP model because of high memory requirements to store nodes of the branch-and-bound tree and high CPU time needed to solve the linear programming relaxation at each node; see Guan et al. (2003), Li and Shahidehpour (2005), Fu et al. (2013), Huang et al. (2017) for a detailed exposure.

The standard BD applied to (1a)–(1l) is composed of a master problem that contains binary variables together with the linearized cost variable, and the subproblems that are defined over continuous variables for fixed value assignments to binary variables. The master problem for this BD formulation is written as

$$\min \left\{ \sum_{i=1}^{n} \sum_{j=1}^{T} \left(c^i_f x^i_j + q^i_j \right) + z \,\middle|\, (1\text{b}) - (1\text{e}),\, (\boldsymbol{x}, \boldsymbol{y}, \bar{\boldsymbol{y}}) \in \{0,1\}^{3nT},\, \boldsymbol{q} \in \mathbb{R}^{nT}_{+},\, z \in [-\Gamma, \Gamma] \right\}, \tag{2}$$

where Γ and $-\Gamma$ are some valid bounds on z induced from the MIP formulation, and z represents the objective value of the following subproblem

$$\min \left\{ \sum_{i=1}^{n} \sum_{j=1}^{T} c^i_g p^i_j \,\middle|\, (1\text{f}) - (1\text{j}),\, (\boldsymbol{p}, \bar{\boldsymbol{p}}) \in \mathbb{R}^{2nT}_{+} \right\}. \tag{3}$$

For the two-stage stochastic UCP described above, master problem (2) represents the first-stage model, and subproblems (3) are defined for each demand scenario realized at the second stage. One way to obtain explicit numerical values for Γ bounds is to minimize/maximize the objective function of (3) over the LP relaxation of (1b)–(1l). Section 4.4 presents computational results for this stochastic problem.

When applying the BD algorithm, at each iteration, an optimal solution of the master problem (2) is found and then fed into the subproblem (3), which provides feasibility or optimality cuts to be added back to the master problem. Since the structure of the above BD formulation conforms to that used in the DD-BD approach, we adopt this model as the basis for our analysis. We show in the next section that through exploiting the DD structure, we can streamline the master problem representation, which results in a superior computational performance.

4.2. DD-BD: Master Problem Formulation

The first step in applying the DD-BD approach of Section 3 to the UCP is to construct a DD that represents the feasible region of the master problem (2). Since this set is defined over binary variables $(\boldsymbol{x}, \boldsymbol{y}, \bar{\boldsymbol{y}})$ together with continuous variables $(\boldsymbol{q}, z)$, a typical DD would contain all five variable types in its arc layers. It turns out that it suffices to model variables $\boldsymbol{x}$ and z only in DD arc layers, while recording the value of other variables through state representations. This projection will allow for a direct application of Algorithm 1, and it will lead to a significant size reduction for DD models.

We present the DD structure for the single-unit case, i.e., $n = 1$. The extension to multi-unit case follows similarly by replicating the DD structure for other units, as they can be considered independently in the master problem. For this reason and to simplify notation, we drop the superscript representing unit number in variables and parameters in the sequel.

The construction of the DD representing (2) is given in Algorithm 2. In view of this algorithm, each node $u \in \mathcal{U}_j$ for $j \in \{1, \ldots, T+1\}$ contains a state value of the form (s^+_u, s^-_u) where s^+_u and s^-_u record the number of time periods passed since the last start-up and shut-down of the unit at time j, respectively. An advantage of this state definition is its *memoryless* property that allows for a direct evaluation of the minimum up/down requirements as well as the logarithmic start-up cost at each time period without the need for backtracking to identify the unit status at previous periods; see the proof of Theorem 3 for a detailed analogy. The initial state value at the root node is set to (∞, ∞) to indicate that the unit is down at the start of the planning timeline and is ready to start up if decided to. The output of Algorithm 2 is a DD $\mathcal{D} = (\mathcal{U}, \mathcal{A}, l(.))$ with $T+1$ arc layers representing variables x_j for $j \in \mathcal{T}$, as well as the continuous variable z at the last layer. Each arc $a \in \mathcal{A}$ has a label l_a that represents the value assignment of its corresponding variable, and a weight w_a that captures the objective function rate of that assignment. The following example illustrates an exact DD for the master problem of an instance of the UCP.

Example 4. Consider a UCP instance with one generator over two time periods with the fixed cost c_f, and the logarithmic start-up cost K_k for k consecutive inactive time periods. Further, suppose that the minimum down-time of the generator is one and the minimum up-time is two. The DD depicted in Figure 5 represents the feasible region of the associated master problem where $-\Gamma$ and Γ are valid bounds for the continuous variable z. In this DD, the dashed and solid arcs represent arc labels with values of zero and one, respectively. In addition, the state value of each node and weight of each arc are given next to them, where K_∞ denotes the cost for the first time that the unit starts up.

The next result shows that the solution set of the equivalence class formed by the DD obtained from Algorithm 2 matches the projection of the feasible region of the master problem (2). In what follows, we refer to a solution $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}, \dot{\bar{\boldsymbol{y}}}, \dot{\boldsymbol{q}}, \dot{z})$ of (2) as *non-redundant* if there is no distinct point $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}, \dot{\bar{\boldsymbol{y}}}, \tilde{\boldsymbol{q}}, \dot{z})$ of (2) with a strictly smaller objective value.

Theorem 3. *Let $\mathcal{D}$ be a DD constructed by Algorithm 2 in relation to a UCP with the master problem (2). Then, a point $(\dot{\boldsymbol{x}}, \dot{z})$ encodes an r-t path of length $\dot{c}$ in the equivalence class formed by $\mathcal{D}$, if and only if this point can be extended to a non-redundant solution $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}, \dot{\bar{\boldsymbol{y}}}, \dot{\boldsymbol{q}}, \dot{z})$ of (2) with objective value $\dot{c}$.* □

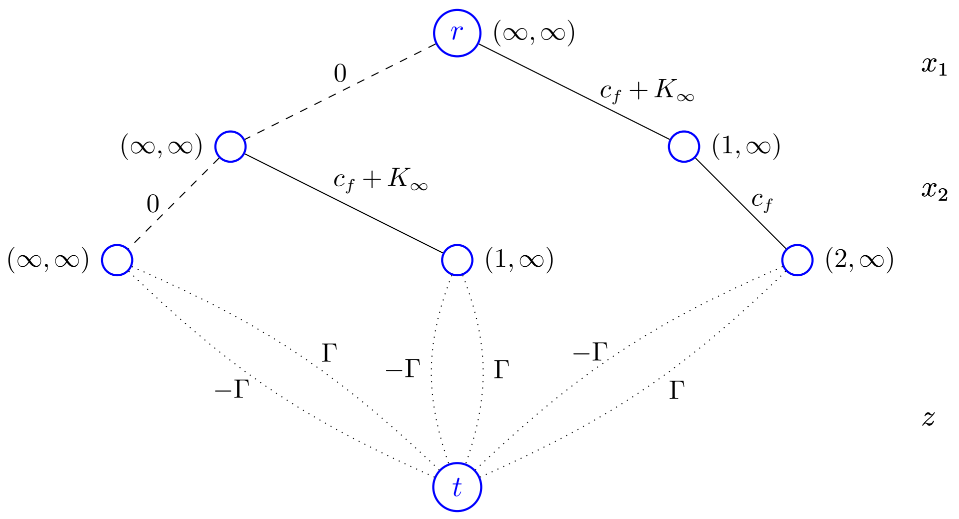


Figure 5 Illustration of an exact DD in Example 4

It follows from Theorem 3 that refining the DD constructed by Algorithm 2 with respect to optimality and feasibility cuts produced from subproblems (3) results in a solution set equivalent to that of the projection of the non-redundant feasible region of (2) after addition of those cuts. As a consequence, the proposed DD-BD approach converges to an optimal solution of the UCP problem. We will give details on the derivation of optimality and feasibility cuts in the next section.

Algorithm 2 demonstrates the construction of an *exact* DD representing the master problem (2), which can be used as the DD oracle required for Algorithm 1. For practical applications, however, the construction of such a DD could be computationally prohibitive. To mitigate this computational burden, as discussed in Section 3, it is necessary to use restricted and relaxed DDs with smaller width limits to improve efficiency of the algorithm. We next give details on designing a relaxed DD for (2).

At the heart of a relaxed DD lies a *node-merging* operation, through which multiple nodes at each layer of the DD are merged into one node, and thereby reducing the DD size. This node-merging operation must satisfy the condition that any r-t path of the exact DD remains an r-t path of the resulting DD with no greater length (for a minimization original problem.) As a byproduct, such a merging operation could create some infeasible r-t paths, hence providing a relaxation. The following definitions set the stage for building a relaxed DD for the UCP.

Algorithm 2: The construction of DD for the master problem of the UCP for $n=1$

```
Data: parameters $\ell$, $L$, $\Gamma$, $K$
Result: a weighted DD $[\mathcal{D}|w(.)]$
1  create the root node with state value $(\infty,\infty)$.
2  forall $j \in \mathcal{T}$ and $u \in \mathcal{U}_j$ do
3      if $s_u^+ \geq s_u^-$ then
4          create a node $v \in \mathcal{U}_{j+1}$ with state value $(s_u^+ + 1, s_u^- + 1)$, and an arc $a \in \mathcal{A}_j$ connecting
            $u$ to $v$ with $l_a = 0$ and $w_a = 0$
5          if $s_u^- \geq \ell$ then
6              create a node $v \in \mathcal{U}_{j+1}$ with state value $(1, s_u^- + 1)$, and an arc $a \in \mathcal{A}_j$ connecting
                $u$ to $v$ with $l_a = 1$ and $w_a = c_f + K_{s_u^-}$
7      else
8          create a node $v \in \mathcal{U}_{j+1}$ with state value $(s_u^+ + 1, s_u^- + 1)$, and an arc $a \in \mathcal{A}_j$ connecting
            $u$ to $v$ with $l_a = 1$ and $w_a = c_f$
9          if $s_u^+ \geq L$ then
10             create a node $v \in \mathcal{U}_{j+1}$ with state value $(s_u^+ + 1, 1)$, and an arc $a \in \mathcal{A}_j$ connecting
                $u$ to $v$ with $l_a = 0$ and $w_a = 0$
11 forall $u \in \mathcal{U}_{T+1}$ do
12     create two arcs $a_1, a_2 \in \mathcal{A}_{T+1}$ connecting $u$ to the terminal node with $l_{a_1} = w_{a_1} = \Gamma$,
        $l_{a_2} = w_{a_2} = -\Gamma$.
```

Definition 2. Consider the following modification of Algorithm 2: (i) each node $u \in \mathcal{U}_j$ for $j \in \{1, \dots, T+1\}$ is assigned with state values $(s_u^+, s_u^-, s_u^=)$ with an additional component $s_u^=$ whose value is initialized and updated similarly to that of s_u^- throughout the iterations of the algorithm; (ii) parameter $K_{s_u^-}$ is replaced with $K_{s_u^=}$ when calculating arc weights in line 6 of the algorithm.

Definition 3. Consider the following node-merging operation defined over the DDs obtained from the algorithm of Definition 2. At layer $j \in \{1, \dots, T+1\}$, a collection of nodes $\{u_1, \dots, u_k\}$ with the property that either $s^+_{u_i} \geq s^-_{u_i}$ for all $i \in \{1, \dots, k\}$, or $s^+_{u_i} < s^-_{u_i}$ for all $i \in \{1, \dots, k\}$ are merged into node v with state values $s^+_v = \max_{i \in \{1,\dots,k\}} s^+_{u_i}$, $s^-_v = \max_{i \in \{1,\dots,k\}} s^-_{u_i}$, and $s^=_v = \min_{i \in \{1,\dots,k\}} s^=_{u_i}$.

We next show that the modified algorithm of Definition 2 combined with the node-merging operation of Definition 3 yield a relaxed DD for the master problem (2).

Proposition 3. *Let $\bar{\mathcal{D}}$ be the DD obtained from Algorithm 2 in relation to the master problem* (2)*. Let $\tilde{\mathcal{D}}$ be a DD constructed from the modified algorithm of Definition 2 in combination with the node-merging operation of Definition 3. Then, $\tilde{\mathcal{D}}$ provides a relaxation of $\bar{\mathcal{D}}$.* □

4.3. DD-BD: Subproblem Formulation

Through application of Algorithm 1 to the UCP, subproblem (3) is solved for fixed values of master variables to obtain feasibility and optimality cuts, with respect to which the restricted and relaxed DDs representing the master problem are refined. These cuts, however, are generated in the space of variable $(\boldsymbol{x}, \boldsymbol{y}, \bar{\boldsymbol{y}}, z)$. As a result, they cannot be directly used to refine the DDs representing the master problem as they only contain variables $(\boldsymbol{x}, z)$; see Section 4.2. To resolve this discrepancy, we next show that constraints (1f) and (1g) in the description of the UCP can be replaced with two constraints that do not contain variables $(\boldsymbol{y}, \bar{\boldsymbol{y}})$. Such a substitution in the subproblem (3) leads to feasibility and optimality cuts that are in the space of $(\boldsymbol{x}, z)$, which can be directly used to refine DDs of the master problem.

Proposition 4. *Assume that $SU^i \leq RU^i$ and $SD^i \leq RD^i$ for all generators $i \in N$. Let $\mathcal{E}$ be the feasible region of the UCP described by* (1b)–(1l)*. Define $\mathcal{G}$ to be the feasible region of the UCP where constraints* (1f) *and* (1g) *are respectively replaced with*

$$p^i_j - p^i_{j-1} \leq \left(RU^i - SU^i\right) x^i_{j-1} + SU^i x^i_j \qquad \forall j \in \mathcal{T}, \quad \forall i \in N \tag{4a}$$

$$p^i_{j-1} - p^i_j \leq \left(RD^i - SD^i\right) x^i_j + SD^i x^i_{j-1} \qquad \forall j \in \mathcal{T}, \quad \forall i \in N. \tag{4b}$$

Then, $\mathcal{E} = \mathcal{G}$. □

Substituting (1f) and (1g) with (4a) and (4b) in the description of the subproblem (3), we obtain the following dual problem

$$\begin{aligned}
\max \quad & \sum_{j=1}^{T} D_j \psi_j + \sum_{j=1}^{T} (D_j + R_j)\, \beta_j + \sum_{i=1}^{n}\sum_{j=1}^{T} \bar{x}_j^i \left(m^i \phi_{i,j} - M^i \pi_{i,j}\right) && (5a)\\
& + \sum_{i=1}^{n}\sum_{j=1}^{T} \Big(\left(SU^i - RU^i\right) \bar{x}_{j-1}^i - SU^i \bar{x}_j^i \Big) \gamma_{i,j} \\
& + \sum_{i=1}^{n}\sum_{j=1}^{T} \Big(\left(SD^i - RD^i\right) \bar{x}_j^i - SD^i \bar{x}_{j-1}^i \Big) \delta_{i,j} \\
\text{s.t.} \quad & \psi_j - \gamma_{i,j} + \gamma_{i,j+1} + \delta_{i,j} - \delta_{i,j+1} + \phi_{i,j} - \eta_{i,j} \le c_g^i \qquad \forall j \in \mathcal{T}, \quad \forall i \in N && (5b)\\
& \beta_j + \eta_{i,j} - \pi_{i,j} \le 0 \qquad \forall j \in \mathcal{T}, \quad \forall i \in N && (5c)\\
& \psi_j, \beta_j, \gamma_{i,j}, \delta_{i,j}, \phi_{i,j}, \eta_{i,j}, \pi_{i,j} \ge 0 \qquad \forall j \in \mathcal{T}, \quad \forall i \in N. && (5d)
\end{aligned}$$

When applying Algorithm 1, we solve the above dual problem for each given assignment $\bar{\boldsymbol{x}}$. If the dual subproblem is unbounded (i.e., the subproblem is infeasible,) we obtain a ray $(\tilde{\boldsymbol{\psi}}, \tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\phi}}, \tilde{\boldsymbol{\pi}}, \tilde{\boldsymbol{\gamma}}, \tilde{\boldsymbol{\delta}})$ and add the following feasibility cut to the master problem.

$$\begin{aligned}
0 \ge & \sum_{j=1}^{T} D_j \tilde{\psi}_j + \sum_{j=1}^{T} (D_j + R_j)\, \tilde{\beta}_j + \sum_{i=1}^{n}\sum_{j=1}^{T} \left(m^i \tilde{\phi}_{i,j} - M^i \tilde{\pi}_{i,j}\right) x_j^i && (6)\\
& + \sum_{i=1}^{n}\sum_{j=1}^{T} \bigg(\tilde{\gamma}_{i,j} \Big(\left(SU^i - RU^i\right) x_{j-1}^i - SU^i x_j^i \Big) + \tilde{\delta}_{i,j} \Big(\left(SD^i - RD^i\right) x_j^i - SD^i x_{j-1}^i \Big) \bigg).
\end{aligned}$$

Otherwise, we obtain an optimal solution $(\boldsymbol{\psi}^*, \boldsymbol{\beta}^*, \boldsymbol{\phi}^*, \boldsymbol{\pi}^*, \boldsymbol{\gamma}^*, \boldsymbol{\delta}^*)$ of the dual subproblem and add the following optimality cut to the master problem.

$$\begin{aligned}
z \ge & \sum_{j=1}^{T} D_j \psi_j^* + \sum_{j=1}^{T} (D_j + R_j)\, \beta_j^* + \sum_{i=1}^{n}\sum_{j=1}^{T} \left(m^i \phi_{i,j}^* - M^i \pi_{i,j}^*\right) x_j^i && (7)\\
& + \sum_{i=1}^{n}\sum_{j=1}^{T} \bigg(\gamma_{i,j}^* \Big(\left(SU^i - RU^i\right) x_{t-j}^i - SU^i x_j^i \Big) + \delta_{i,j}^* \Big(\left(SD^i - RD^i\right) x_j^i - SD^i x_{j-1}^i \Big) \bigg).
\end{aligned}$$

4.4. Computational Experiments

In this section, we assess the performance of the proposed DD-BD algorithm in solving the stochastic UCP instances for a short-term (day-ahead) power generation model. For our experiments, we consider benchmark instances from the problem class OR-LIB/UC in UnitCommitment.jl (Xavier et al. 2020). Since these instances lack some of the modeling elements in the UCP variant we consider in this paper, we set their associated parameter values as follows. We set the production cost c_g^i for unit $i \in N$ as the average of production costs for that unit. We create the fixed operating cost c_f^i for each unit $i \in N$ randomly from the interval $[400, 1000]$. We set $RU^i = SU^i$ and $RD^i = SD^i$ for each unit $i \in N$. We capture demand uncertainty by a set of scenarios Ξ that are generated randomly from the interval $[0.75\bar{M}, \bar{M}]$ where $\bar{M}$ denotes the total capacity over all generators. All experiments were conducted on a machine running Windows 10, x64 operating system with Intel® Core i7 processor (2.60 GHz) and 32 GB RAM. The code is written in Microsoft Visual Studio 2015 in C++. For the optimization parts, we use the Gurobi solver version 9.0.0.

4.4.1. Comparison with Decomposition Methods We first present computational results that compare the outcome of the DD-BD method, as a decomposition technique, with other standard decomposition methods. In particular, we consider two BD approaches referred to as G-BD 1 and G-BD 2.

G-BD 1 employs the classical BD in a *textbook* fashion, where the master problem (2) is solved by Gurobi at its default settings, and the optimality/feasibility cuts are produced by solving the subproblems (3) for each demand scenario.

Since the traditional implementation of BD, as used in G-BD 1, is known to suffer from a slow convergence, we employ modern modeling and algorithmic practices in the literature that boost the performance of the G-BD. We refer to this improved approach as G-BD 2. The details of these improvements are as follows. (i) To accelerate the convergence rate associated with the feasibility cuts, we use the L^1 normalization technique proposed by by Fischetti et al. (2010) and Bonami et al. (2020). Consider the MIP $\mathcal{H}$ defined in Section 3. Assume that the subproblem $\mathcal{S}(\bar{\boldsymbol{x}})$ is infeasible for a fixed solution $\bar{\boldsymbol{x}}$ obtained from the master problem. It follows that $w^* = \min_{w,\boldsymbol{y}}\{w \mid B\boldsymbol{y} + w\mathbf{1} \leq \boldsymbol{c} - A\bar{\boldsymbol{x}}\} = \boldsymbol{\pi}^*(\boldsymbol{c} - A\bar{\boldsymbol{x}}) > 0$, where $\boldsymbol{\pi}^*$ is the optimal dual vector of the *normalized* subproblem, $\mathbf{1}$ is a unit column vector, and the equality holds because of the strong duality property. Therefore, the inequality $\boldsymbol{\pi}^*(\boldsymbol{c} - A\bar{\boldsymbol{x}}) \leq 0$ can be added to the master problem as a feasibility cut that excludes $\bar{\boldsymbol{x}}$.

(ii) During presolve, we generate heuristic cuts to tighten the initial LP relaxation of the master problem as follows. At each time period, we identify the scenario with the highest demand and add valid inequalities in the space of commitment variables to ensure the feasibility of constraints (1h) and (1j). (iii) We derive the so-called *combinatorial cuts* that exclude the current infeasible solution of the master problem by forcing a change of value in at least one commitment variable; see Codato and Fischetti (2006) and Rodríguez et al. (2021) for details about these cuts. (iv) Finally, we strengthen the logarithmic start-up cost constraints (1b) by reducing the coefficient of variables x^i_{j-h} as proposed in Silbernagl et al. (2015) and Knueven et al. (2020a).

For the DD-BD method, we set the width limit for the restricted/relaxed DDs to two and use the node-merging operation given in Definition 3 for the construction of relaxed DDs. When necessary during the refinement procedures, we allow the increase in the DD width to accommodate for the separation of current non-optimal solutions.

For these experiments, we fix the number of demand scenarios at 100. We consider four size categories with $n \in \{20, 50, 100, 150\}$ for the number of generators, and solve five 5 instances for each problem size as given in the OR-LIB/UC. Table 1 contains the result of solving these instances with G-BD 1, G-BD 2 and DD-BD. For these runs, we fix the time limit at 7200 seconds. The first column of Table 1 indicates the number of generators. The second column shows the instance number. The columns under "G-BD 1" report the solution time (in seconds), the total number of generated feasibility and optimality cuts during the solution process. The subsequent columns contain the same quantities for G-BD 2 and DD-BD. As observed in Table 1, while the boosting methods implemented in G-BD 2 have substantially improved the solution time over G-BD 1, the DD-BD approach uniformly outperforms both G-BD alternatives across all size categories, rendering it as the most effective decomposition approach.

4.4.2. Comparison with Full Formulations In this section, we compare the performance of the decomposition methods DD-BD and G-BD 2 with that of the full formulation (1a)–(1l) when solved by Gurobi at its default settings. Since it is well-known that decomposition methods are

Table 1 Solution times (in seconds) of G-BD 1, G-BD 2, and DD-BD for instances of the stochastic UCP with $|\Xi| = 100$.

		G-BD 1			G-BD 2			DD-BD		
n	Instance	time	f cuts	o cuts	time	f cuts	o cuts	time	f cuts	o cuts
20	1	753.21	633	34	527.76	393	31	**381.78**	301	26
20	2	981.60	638	26	508.32	438	28	**393.94**	472	34
20	3	854.45	920	20	499.68	586	19	**371.70**	560	24
20	4	1516.88	196	31	732.96	162	29	**462.88**	183	21
20	5	1240.28	264	34	805.59	178	33	**518.86**	121	25
50	1	4633.54	1119	73	2683.69	939	65	**1238.59**	663	79
50	2	4094.20	973	60	2154.84	850	54	**1318.84**	731	40
50	3	3791.37	1428	33	2071.47	899	32	**1149.24**	836	29
50	4	3571.56	904	19	1623.54	554	18	**1105.84**	459	21
50	5	4002.24	1023	25	1589.73	673	23	**1179.27**	483	19
100	1	> 7200	1917	29	4731.48	1464	78	**3430.76**	1009	72
100	2	> 7200	2532	21	3956.04	1576	36	**3038.67**	1162	26
100	3	> 7200	2187	20	4121.46	1257	47	**3267.31**	1206	36
100	4	> 7200	1375	43	4823.22	1042	56	**3180.45**	794	42
100	5	> 7200	1967	19	3781.28	1272	31	**2688.28**	864	37
150	1	> 7200	1688	32	6944.31	1110	94	**5697.86**	1043	83
150	2	> 7200	1803	23	7019.38	1097	87	**6246.27**	976	79
150	3	> 7200	1501	21	6759.04	1065	53	**5393.86**	979	40
150	4	> 7200	1350	39	6491.46	993	55	**5202.67**	744	65
150	5	> 7200	1635	33	6826.97	1046	81	**5850.57**	920	74

advantageous for larger problem sizes, we conduct these experiments for an increased number of demand scenarios, i.e., $|\Xi| \in \{200, 400, 600, 800\}$, to have a meaningful comparison. These results are given in Tables 2 and 3 for 20 and 50 generators, respectively. The solution time for larger number of generators exceeds the time limit of 7200 seconds for all methods. In these tables, the first column shows the number of demand scenarios, and the second column contains the instance number. The next three columns under "G-BD 2" report the solution time, the number of feasibility and optimality cuts, respectively. Columns 6-8 contain similar results for the DD-BD method. The last column shows the solution time for the full formulation.

As evident in Tables 2 and 3, similarly to Table 1, the DD-BD method outperforms the G-BD 2 uniformly for all problem sizes. Even though the full formulation solves the instances with 200 and 400 demand scenarios faster than the decomposition methods, it suffers from a steeper time increase as the number of demand scenarios increases. As a result, the DD-BD method catches up to the full formulation for 600 demand scenarios and overcomes it for the larger size with 800 demand scenarios, implying the superior performance of the DD-BD compared to other approaches for large problems sizes. To better visualize this pattern, we present the average solution time for each size category against the number of demand scenarios in Figures 6 and 7.

We conclude this section by noting that the presented computational results are obtained through a basic implementation of the DD-BD without incorporating any computational enhancement methods for DDs such as *variables ordering*, *DD reduction*, *dynamic width adjustment*, *primal heuristics*, etc. As common in the literature, such basic DD implementations are often compared against modern solvers with the presolve/cuts/heuristics settings turned off to provide a fair comparison for the core solution methodology. Despite being at such disadvantage, the above results indicate that the proposed DD-BD method can still outperform the G-BD and the full UCP formulation solved with the solver's default settings. While this shows the potential of the DD-BD method in energy applications, its performance can be further improved using modern computational practices in the DD literature, which can be considered as a direction for future work.

Table 2 Solution times (in seconds) of G-BD 2, DD-BD, and Full formulations for instances of the stochastic UCP with $n = 20$

		G-BD 2			DD-BD			Full
$\lvert\Xi\rvert$	Instance	time	f cuts	o cuts	time	f cuts	o cuts	time
200	1	945.18	452	42	708.96	308	38	**304.05**
200	2	1004.18	562	31	542.99	385	33	**186.35**
200	3	1186.93	612	31	510.94	531	36	**252.89**
200	4	809.82	550	42	537.76	379	44	**290.98**
200	5	707.68	599	31	650.80	515	23	**371.19**
400	1	1618.80	1034	52	946.88	953	59	**616.48**
400	2	1812.31	847	42	856.44	695	35	**574.28**
400	3	1451.45	514	38	1025.23	395	32	**598.08**
400	4	1158.77	564	58	942.89	600	41	**606.82**
400	5	1696.87	712	54	721.38	575	61	**618.77**
600	1	2115.54	909	56	1357.43	736	54	**1132.66**
600	2	1530.90	1163	53	**1338.87**	1314	48	1413.65
600	3	1826.37	1241	60	1270.83	1024	48	**1142.26**
600	4	2190.60	1226	80	**1353.67**	875	89	1488.27
600	5	1950.40	984	68	1256.71	728	50	**1192.87**
800	1	2879.51	1410	76	**2231.36**	1049	61	2601.54
800	2	2332.64	1119	70	**2195.52**	758	83	2653.63
800	3	2651.88	1458	71	**2254.60**	1107	86	2921.57
800	4	2958.79	1623	88	**2288.74**	1671	75	2772.84
800	5	2707.54	1570	86	**2635.46**	1522	69	2897.46

Table 3 Solution times (in seconds) of G-BD 2, DD-BD, and Full formulations for instances of the stochastic UCP with $n = 50$

		G-BD 2			DD-BD			Full
$\|\Xi\|$	Instance	time	f cuts	o cuts	time	f cuts	o cuts	time
200	1	3505.12	1643	78	1944.37	1549	90	**610.70**
200	2	3797.28	1066	43	1816.58	970	47	**712.24**
200	3	2824.65	972	30	2210.88	916	22	**628.35**
200	4	2949.56	855	20	2034.54	664	15	**850.44**
200	5	3375.60	1478	84	1981.47	1028	76	**773.09**
400	1	3859.64	1804	70	3324.22	1629	75	**1752.74**
400	2	4526.40	860	72	3092.76	621	52	**1939.73**
400	3	4129.86	1440	58	3920.40	1209	49	**1658.43**
400	4	4653.48	1776	38	2971.42	1633	41	**1463.31**
400	5	4367.37	1383	81	3656.57	1211	100	**1277.31**
600	1	5785.03	1251	109	**4003.92**	1075	91	4498.25
600	2	5834.85	1537	99	4583.70	1618	87	**4124.68**
600	3	5410.30	1401	60	**4970.63**	1162	51	5065.13
600	4	5384.34	1950	53	**4409.66**	1482	62	4874.60
600	5	5739.52	1136	89	5094.68	1056	67	**5048.91**
800	1	6396.84	2285	94	**5963.94**	1713	74	6883.02
800	2	6575.52	2303	110	**6204.16**	1704	129	6840.31
800	3	6606.54	1831	82	**6353.15**	1617	76	6643.29
800	4	6861.92	1161	71	**6065.85**	940	59	6411.49
800	5	6919.75	2843	107	**6182.09**	1998	121	6757.77

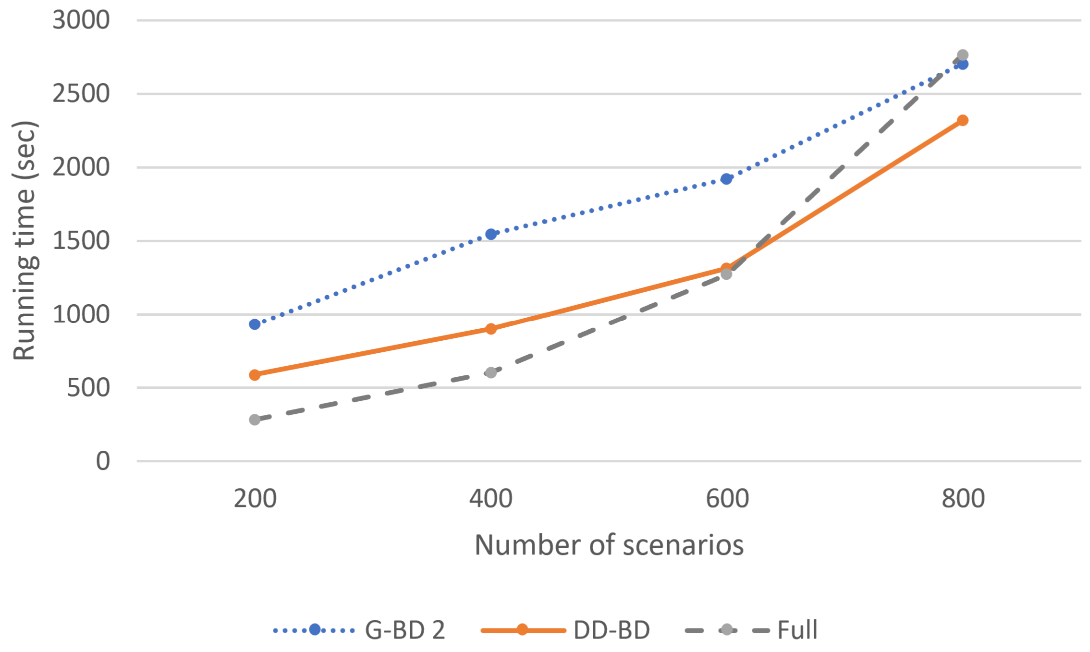


Figure 6 Average solution times of the G-BD 2, DD-BD, and Full formulations for different number of scenarios and $n = 20$

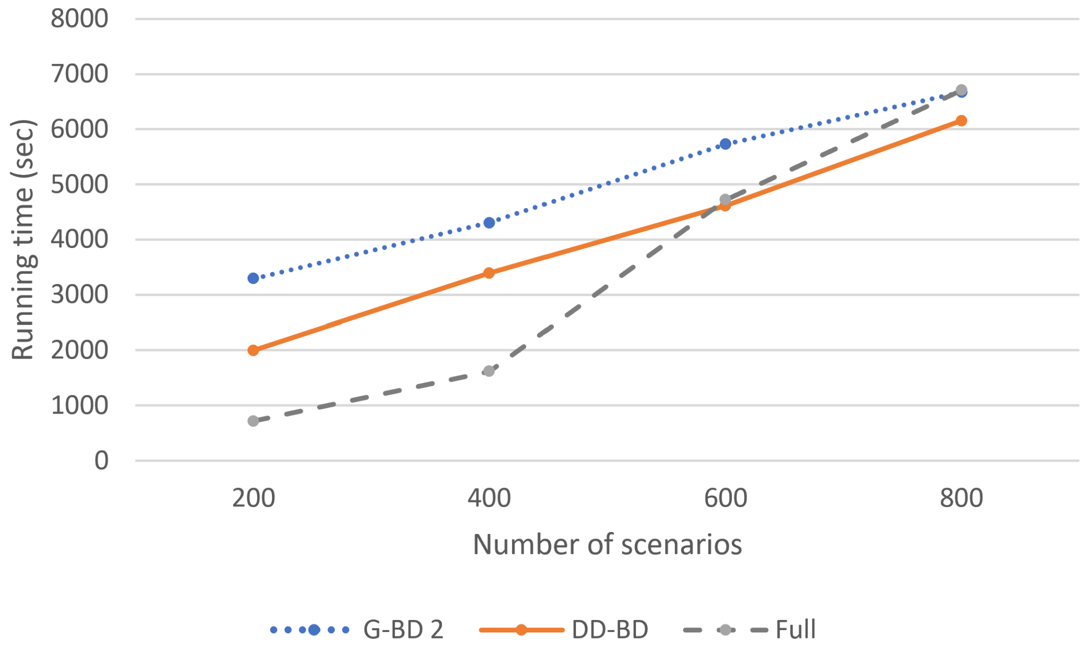


Figure 7 Average solution times of the G-BD 2, DD-BD, and Full formulations for different number of scenarios and $n = 50$

5. Conclusion

In this paper, we propose a geometric decomposition that restructures a mixed integer set through a collection of hyper-rectangle formations. We show that this decomposition approach is the key to model optimization problems with both integer and continuous variables through decision diagrams. In this regard, we introduce a method, referred to as DD-BD, that extends the applicability domain of decision diagrams to MIPs. We evaluate the performance of DD-BD by applying it to the unit commitment problem in the electric grid market.

Acknowledgement

This project is sponsored in part by the Iowa Energy Center, Iowa Economic Development Authority and its utility partners. We thank the anonymous referees and the Associate Editor for their helpful comments that contributed to improving the paper.

References

Atakan S, Lulli G, Sen S (2017) A state transition mip formulation for the unit commitment problem. *IEEE Transactions on Power Systems* 33(1):736–748.

Baldick R (1995) The generalized unit commitment problem. *IEEE Transactions on Power Systems* 10(1):465–475.

Bendotti P, Fouilhoux P, Rottner C (2018) The min-up/min-down unit commitment polytope. *Journal of Combinatorial Optimization* 36(3):1024–1058.

Bendotti P, Fouilhoux P, Rottner C (2019) On the complexity of the unit commitment problem. *Annals of Operations Research* 274(1-2):119–130.

Bergman D, Cire AA (2016) Multiobjective optimization by decision diagrams. *International Conference on Principles and Practice of Constraint Programming*, 86–95 (Springer).

Bergman D, Cire AA (2018) Discrete nonlinear optimization by state-space decompositions. *Management Science* 64(10):4700–4720.

Bergman D, Cire AA, van Hoeve WJ (2015) Lagrangian bounds from decision diagrams. *Constraints* 20(3):346–361.

Bergman D, Cire AA, van Hoeve WJ, Hooker J (2016) *Decision Diagrams for Optimization* (Springer International Publishing).

Bertsimas D, Litvinov E, Sun XA, Zhao J, Zheng T (2012) Adaptive robust optimization for the security constrained unit commitment problem. *IEEE transactions on power systems* 28(1):52–63.

Blanco I, Morales JM (2017) An efficient robust solution to the two-stage stochastic unit commitment problem. *IEEE Transactions on Power Systems* 32(6):4477–4488.

Bonami P, Salvagnin D, Tramontani A (2020) Implementing automatic benders decomposition in a modern mip solver. *International conference on integer programming and combinatorial optimization*, 78–90 (Springer).

Carrión M, Arroyo JM (2006) A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. *IEEE Transactions on power systems* 21(3):1371–1378.

Codato G, Fischetti M (2006) Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research* 54(4):756–766.

Conforti M, Cornuéjols G, Zambelli G (2014a) *Integer Programming* (Springer).

Conforti M, Cornuéjols G, Zambelli G, et al. (2014b) *Integer programming*, volume 271 (Springer).

Davarnia D (2021) Strong relaxations for continuous nonlinear programs based on decision diagrams. *Operations Research Letters* URL `http://dx.doi.org/10.1016/j.orl.2021.01.011`.

Davarnia D, van Hoeve WJ (2020) Outer approximation for integer nonlinear programs via decision diagrams. *Mathematical Programming* URL `http://dx.doi.org/10.1007/s10107-020-01475-4`.

Feng Y, Ryan SM (2016) Solution sensitivity-based scenario reduction for stochastic unit commitment. *Computational Management Science* 13(1):29–62.

Fischetti M, Salvagnin D, Zanette A (2010) A note on the selection of benders' cuts. *Mathematical Programming* 124(1):175–182.

Frangioni A, Gentile C (2006) Solving nonlinear single-unit commitment problems with ramping constraints. *Operations Research* 54(4):767–775.

Frangioni A, Gentile C, Lacalandra F (2008) Tighter approximated milp formulations for unit commitment problems. *IEEE Transactions on Power Systems* 24(1):105–113.

Franz A, Rieck J, Zimmermann J (2020) A long-term unit commitment problem with hydrothermal coordination for economic and emission control in large-scale electricity systems. *OR spectrum* 42(1):235–259.

Fu Y, Li Z, Wu L (2013) Modeling and solution of the large-scale security-constrained unit commitment. *IEEE Transactions on Power Systems* 28(4):3524–3533.

Garver LL (1962) Power generation scheduling by integer programming-Development of theory. *Transactions of the American Institute of Electrical Engineers. Part III: Power Apparatus and Systems* 81(3):730–734.

Gonzalez J, Cire A, Lodi A, Rousseau LM (2020) Integrated integer programming and decision diagram search tree with an application to the maximum independent set problem. *Constraints* 25:23–46.

Guan X, Zhai Q, Papalexopoulos A (2003) Optimization based methods for unit commitment: Lagrangian relaxation versus general mixed integer programming. *2003 IEEE Power Engineering Society General Meeting (IEEE Cat. No. 03CH37491)*, volume 2, 1095–1100 (IEEE).

Hadžić T, Hooker JN (2006) Discrete global optimization with binary decision diagrams. *Workshop on Global Optimization: Integrating Convexity, Optimization, Logic Programming, and Computational Algebraic Geormetry (GICOLAG).*

Huang Y, Pardalos PM, Zheng QP (2017) *Electrical power unit commitment: deterministic and two-stage stochastic programming models and algorithms* (Springer).

Knueven B, Ostrowski J, Watson JP (2020a) A novel matching formulation for startup costs in unit commitment. *Mathematical Programming Computation* 1–24.

Knueven B, Ostrowski J, Watson JP (2020b) On mixed-integer programming formulations for the unit commitment problem. *INFORMS Journal on Computing* .

Lee FN, Feng Q (1992) Multi-area unit commitment. *IEEE Transactions on power systems* 7(2):591–599.

Li T, Shahidehpour M (2005) Price-based unit commitment: A case of lagrangian relaxation versus mixed integer programming. *IEEE transactions on power systems* 20(4):2015–2025.

Li Y, McCalley JD, Ryan S (2007) Risk-based unit commitment. *2007 IEEE Power Engineering Society General Meeting*, 1–7 (IEEE).

Lorca Á, Sun XA, Litvinov E, Zheng T (2016) Multistage adaptive robust optimization for the unit commitment problem. *Operations Research* 64(1):32–51.

Maifeld TT, Sheble GB (1996) Genetic-based unit commitment algorithm. *IEEE Transactions on Power systems* 11(3):1359–1370.

Mantawy A, Abdel-Magid YL, Selim SZ (1998) A simulated annealing algorithm for unit commitment. *IEEE Transactions on Power Systems* 13(1):197–204.

Morales-España G, Latorre JM, Ramos A (2013) Tight and compact milp formulation for the thermal unit commitment problem. *IEEE Transactions on Power Systems* 28(4):4897–4908.

Nemhauser GL, Wolsey LA (1990) A recursive procedure to generate all cuts for 0–1 mixed integer programs. *Mathematical Programming* 46(1-3):379–390.

Ostrowski J, Anjos MF, Vannelli A (2011) Tight mixed integer linear programming formulations for the unit commitment problem. *IEEE Transactions on Power Systems* 27(1):39–46.

Padhy NP (2004) Unit commitment-a bibliographical survey. *IEEE Transactions on power systems* 19(2):1196–1205.

Pang C, Sheblé GB, Albuyeh F (1981) Evaluation of dynamic programming based methods and multiple area representation for thermal unit commitments. *IEEE Transactions on Power Apparatus and Systems* (3):1212–1218.

Queyranne M, Wolsey LA (2017) Tight mip formulations for bounded up/down times and interval-dependent start-ups. *Mathematical Programming* 164(1-2):129–155.

Rajan CA, Mohan M, Manivannan K (2003) Neural-based tabu search method for solving unit commitment problem. *IEE Proceedings-Generation, Transmission and Distribution* 150(4):469–474.

Rajan D, Takriti S, et al. (2005) Minimum up/down polytopes of the unit commitment problem with start-up costs. *IBM Res. Rep* 23628:1–14.

Rockafeller RT (1970) *Convex Analysis* (New Jersey, NJ: Princeton University Press).

Rodríguez JA, Anjos MF, Côté P, Desaulniers G (2021) Accelerating benders decomposition for short-term hydropower maintenance scheduling. *European Journal of Operational Research* 289(1):240–253.

Saneifard S, Prasad NR, Smolleck HA (1997) A fuzzy logic approach to unit commitment. *IEEE Transactions on Power Systems* 12(2):988–995.

Serra T, Hooker J (2020) Compact representation of near-optimal integer programming solutions. *Mathematical Programming* 182:199–232.

Silbernagl M, Huber M, Brandenberg R (2015) Improving accuracy and efficiency of start-up cost formulations in mip unit commitment by modeling power plant temperatures. *IEEE Transactions on Power Systems* 31(4):2578–2586.

Tuffaha M, Gravdahl JT (2013) Mixed-integer formulation of unit commitment problem for power systems: Focus on start-up cost. *IECON 2013-39th Annual Conference of the IEEE Industrial Electronics Society*, 8160–8165 (IEEE).

van Ackooij W, Danti Lopez I, Frangioni A, Lacalandra F, Tahanan M (2018) Large-scale unit commitment under uncertainty: an updated literature survey. *Annals of Operations Research* 271(1):11–85.

van der Linden K (2017) Decision diagrams for decomposed mixed integer linear programs. Master's thesis. Delft University of Technology.

Wu L (2011) A tighter piecewise linear approximation of quadratic cost curves for unit commitment problems. *IEEE Transactions on Power Systems* 26(4):2581–2583.

Xavier AS, Kazachkov AM, Qiu F (2020) Unitcommitment.jl: A julia/jump optimization package for security-constrained unit commitment. Zenodo (2020). DOI: 10.5281/zenodo.4269874.

Zheng QP, Wang J, Liu AL (2014) Stochastic optimization for unit commitment—a review. *IEEE Transactions on Power Systems* 30(4):1913–1924.

Zheng QP, Wang J, Pardalos PM, Guan Y (2013) A decomposition approach to the two-stage stochastic unit commitment problem. *Annals of Operations Research* 210(1):387–410.

Appendix A: Omitted Proofs

Proof of Theorem 1. For a given function $f(\boldsymbol{x})$ that is convex in $\boldsymbol{x}_I$, define $z^1 = \max\{f(\boldsymbol{x})|\boldsymbol{x} \in \bigcup_{j\in J} \mathcal{X}(P_I^j)\}$ and $z^2 = \max\{f(\boldsymbol{x})|\boldsymbol{x} \in \bigcup_{j\in J}\bigcup_{k\in K_j} R_j^k\}$. We first show that $z^1 = z^2$. For each $j \in J$, we write that $\mathcal{X}(P_I^j) \subseteq \mathcal{X}(\bigcup_{k\in K_j} R_j^k) \subseteq \bigcup_{k\in K_j} R_j^k$, where the first inclusion follows from condition (iii), and the second inclusion follows from definition of extreme points. Therefore, $z^1 \leq z^2$. The above inclusions also show that $\bigcup_{j\in J} \mathcal{X}(P_I^j)$ is a finite set since $\bigcup_{j\in J} \mathcal{X}(\bigcup_{k\in K_j} R_j^k)$ is finite because of the finiteness of J, K_j and the set of extreme points of hyper-rectangles R_j^k. Let $\boldsymbol{x}^*$ be an optimal solution for z^2. Such optimal solution exists because of the compactness of the feasible region. It follows that $\boldsymbol{x}^* \in \bigcup_{k\in K_{j^*}} R_{j^*}^k$ for some $j^* \in J$. Condition (i) implies that coordinates $i \in N \setminus I$ are fixed in $P_I^{j^*}$, and condition (iii) implies that these coordinates must also be fixed for $\bigcup_{k\in K_{j^*}} R_{j^*}^k$. Therefore, x_i^* is fixed at coordinates $i \in N \setminus I$. Since $f(\boldsymbol{x})$ is convex in the unfixed variables $\boldsymbol{x}_I$, its maximum over $\bigcup_{k\in K_{j^*}} R_{j^*}^k$ occurs at an extreme point $\bar{\boldsymbol{x}}$ of $\text{conv}(\bigcup_{k\in K_{j^*}} R_{j^*}^k)$. Condition (iii) implies that $\bar{\boldsymbol{x}} \in \mathcal{X}(P_I^{j^*})$, proving that $z^1 \geq z^2$. For the second part of the proof, we show that $z^3 = \max\{f(\boldsymbol{x})|\boldsymbol{x} \in \mathcal{P}\} = z^1 = z^2$. It follows from condition (ii) that $z^1 \leq z^3 \leq z^4$ where $z^4 = \max\{f(\boldsymbol{x})|\boldsymbol{x} \in \bigcup_{j\in J} P_I^j\}$. Using an argument similar to that given above, we conclude that the optimal value z^4 is attained at an extreme point of $P_I^{j^*}$ for some $j^* \in J$, which is also an extreme point of $\bigcup_{k\in K_{j^*}} R_{j^*}^k$. Therefore, using the definition of z^4 and z^2, we write that $z^4 \leq z^2 = z^1$, proving the result. □

Proof of Proposition 1. Consider any point $\bar{\boldsymbol{x}}_{N\setminus I} \in \text{proj}_{x_{N\setminus I}}(\mathcal{P})$, and define the indicator function

$$\delta(\boldsymbol{x}|\bar{\boldsymbol{x}}_{N\setminus I}) = \begin{cases} 0, & \text{if } \boldsymbol{x}_{N\setminus I} = \bar{\boldsymbol{x}}_{N\setminus I} \\ -\infty, & \text{else.} \end{cases}$$

We can use $\delta(\boldsymbol{x}|\bar{\boldsymbol{x}}_{N\setminus I})$ as the objective function in the relation $\max\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{P}\} = \max\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{Q}\}$ as it is convex in $\boldsymbol{x}_I$. This follows from the fact that $\delta(\boldsymbol{x}|\bar{\boldsymbol{x}}_{N\setminus I}) = 0$ when $\boldsymbol{x}_{N\setminus I} = \bar{\boldsymbol{x}}_{N\setminus I}$, and it is $-\infty$ otherwise, which is *improper* convex; see Rockafeller (1970). We conclude that $\bar{\boldsymbol{x}}_{N\setminus I} \in \text{proj}_{x_{N\setminus I}}(\mathcal{Q})$. Using a similar argument for the reverse direction, we conclude that $\text{proj}_{x_{N\setminus I}}(\mathcal{P}) = \text{proj}_{x_{N\setminus I}}(\mathcal{Q})$. This also shows that $\text{proj}_{x_{N\setminus I}}(\mathcal{P})$ is finite as $\mathcal{Q}$ is finite by assumption. For any point $\bar{\boldsymbol{x}}_{N\setminus I} \in \text{proj}_{x_{N\setminus I}}(\mathcal{P})$, define $\mathcal{P}(\bar{\boldsymbol{x}}_{N\setminus I}) = \mathcal{P} \cap \{\boldsymbol{x}|\boldsymbol{x}_{N\setminus I} = \bar{\boldsymbol{x}}_{N\setminus I}\}$, and $\mathcal{Q}(\bar{\boldsymbol{x}}_{N\setminus I}) = \mathcal{Q} \cap \{\boldsymbol{x}|\boldsymbol{x}_{N\setminus I} = \bar{\boldsymbol{x}}_{N\setminus I}\}$. For any convex function $f_I(\boldsymbol{x}_I)$ defined in the space of variables $\boldsymbol{x}_I$, and any point $\bar{\boldsymbol{x}}_{N\setminus I} \in \text{proj}_{x_{N\setminus I}}(\mathcal{P})$, we have that

$$\max\{f_I(\boldsymbol{x}_I)|\boldsymbol{x} \in \mathcal{P}(\bar{\boldsymbol{x}}_{N\setminus I})\} \tag{8a}$$

$$= \max\{\delta(\boldsymbol{x}|\bar{\boldsymbol{x}}_{N\setminus I}) + f_I(\boldsymbol{x}_I)|\boldsymbol{x} \in \mathcal{P}\} \tag{8b}$$

$$= \max\{\delta(\boldsymbol{x}|\bar{\boldsymbol{x}}_{N\setminus I}) + f_I(\boldsymbol{x}_I)|\boldsymbol{x} \in \mathcal{Q}\} \tag{8c}$$

$$= \max\{f_I(\boldsymbol{x}_I)|\boldsymbol{x} \in \mathcal{Q}(\bar{\boldsymbol{x}}_{N\setminus I})\}, \tag{8d}$$

where (8a) and (8d) follow from the definition of the indicator function above, and (8b) follows from the assumption. Since $f_I(\boldsymbol{x}_I)$ is convex, the optimal value of (8a) is attained at an extreme point $\hat{\boldsymbol{x}}$ of $\mathcal{P}(\bar{\boldsymbol{x}}_{N\setminus I})$. For each such extreme point, there are infinitely many convex functions $\hat{f}_I(\boldsymbol{x}_I)$ with $\hat{\boldsymbol{x}}$ as their unique maximizer over $\mathcal{P}(\bar{\boldsymbol{x}}_{N\setminus I})$. For every one of these functions, according to the chain equalities (8a)–(8d), $\mathcal{Q}$ has a point that matches the optimal value. Since $\mathcal{Q}$ is finite, it must be that $\hat{\boldsymbol{x}} \in \mathcal{Q}(\bar{\boldsymbol{x}}_{N\setminus I})$. This shows that the set of extreme points of $\mathcal{P}(\bar{\boldsymbol{x}}_{N\setminus I})$ is finite. Similarly, it can be shown that for any extreme point $\dot{\boldsymbol{x}}$ of $\text{conv}(\mathcal{Q}(\bar{\boldsymbol{x}}_{N\setminus I}))$, we have that $\dot{\boldsymbol{x}} \in \mathcal{P}(\bar{\boldsymbol{x}}_{N\setminus I})$. As a result, $\text{conv}(\mathcal{P}(\bar{\boldsymbol{x}}_{N\setminus I})) = \text{conv}(\mathcal{Q}(\bar{\boldsymbol{x}}_{N\setminus I}))$ for every $\bar{\boldsymbol{x}}_{N\setminus I} \in \text{proj}_{x_{N\setminus I}}(\mathcal{P})$. Now, construct sets P_I^j in Theorem 1 as $\text{conv}(\mathcal{P}(\bar{\boldsymbol{x}}_{N\setminus I}))$ for every $\bar{\boldsymbol{x}}_{N\setminus I} \in \text{proj}_{x_{N\setminus I}}(\mathcal{P})$, which yields a finite collection. Condition (i) is satisfied as each set P_I^j is restricted at $\{\boldsymbol{x} | \boldsymbol{x}_{N\setminus I} = \bar{\boldsymbol{x}}_{N\setminus I}\}$. Condition (ii) holds since for any extreme point of P_I^j, there is a point of $\mathcal{P}$ by construction, and since any point $\bar{\boldsymbol{x}} \in \mathcal{P}$ satisfies $\bar{\boldsymbol{x}} \in \mathcal{P}(\bar{\boldsymbol{x}}_{N\setminus I}) \subseteq \text{conv}(\mathcal{P}(\bar{\boldsymbol{x}}_{N\setminus I}))$. For condition (iii), rectangles R_j^k can be considered as the set of extreme points of P_I^j, which has been shown to be finite. □

Proof of Lemma 1. The result is obtained as a special case of Theorem 1, where sets P_I^j and R_j^k coincide, i.e., $P_I^j = R_j^1$ for all $j \in J$. The conditions and the result follow immediately. □

Proof of Proposition 2. For a given DD $\mathcal{D}$, define a node-sequence as an ordered set of connected nodes from the root to the terminal, i.e., $\boldsymbol{u} = (u_1, u_2, \cdots, u_{n+1})$ where $u_i \in \mathcal{U}_i$ for $i \in N \cup \{n+1\}$. Let U be the collection of all node-sequences of $\mathcal{D}$. For $\boldsymbol{u} \in U$, define

$$S_I(\boldsymbol{u}) = \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} l^{\min}_{(u_i,u_{i+1})} \le x_i \le l^{\max}_{(u_i,u_{i+1})}, & \forall i \in I \\ x_i = l_{(u_i,u_{i+1})}, & \forall i \in N \setminus I \end{array} \right\}.$$

Viewing $\text{Sol}(\mathcal{D})$ as a compact set $\mathcal{P}$ in Lemma 1, it is straightforward to verify that sets $S_I(\boldsymbol{u})$ satisfy the conditions for hyper-rectangles R_I^j. It also follows from the definition of (virtual) DDs that $\text{Sol}(\hat{\mathcal{D}}) = \bigcup_{\boldsymbol{u} \in U} S_I(\boldsymbol{u})$ and $\text{Sol}(\bar{\mathcal{D}}) = \bigcup_{\boldsymbol{u} \in U} \mathcal{X}(S_I(\boldsymbol{u}))$. The result follows from Lemma 1. □

Proof of Corollary 1. For the direct implication, assume that a given compact set $\mathcal{P}$ admits a rectangular decomposition w.r.t. I through sets P_I^j for $j \in J$. Then, the DD that encodes the finite collection of points $\bigcup_{j \in J} \mathcal{X}(P_I^j)$ provides the desired DD representation because of Theorem 1. For the reverse implication, assume that a given compact set $\mathcal{P}$ is DD-representable w.r.t. I through a DD $\mathcal{D}$. Since $\text{Sol}(\mathcal{D})$ is finite, it follows from Proposition 1 that $\mathcal{P}$ admits a rectangular decomposition w.r.t. I. □

Proof of Corollary 2. Since $\mathcal{Q}$ is bounded, there are finitely many points $\bar{\boldsymbol{x}} \in \text{proj}_x(\mathcal{P})$. For each such point, it follows from the definition of projection and the boundedness of $\mathcal{Q}$ that there exists an interval $[l_{\bar{x}}, u_{\bar{x}}]$ such that $(\bar{\boldsymbol{x}}; \bar{y}) \in \mathcal{P}$ for every $\bar{y} \in [l_{\bar{x}}, u_{\bar{x}}]$. As a result, we can write that

$\mathcal{P} = \bigcup_{\bar{\boldsymbol{x}} \in \text{proj}_x(\mathcal{P})} R_y^{\bar{x}}$ where $R_y^{\bar{x}} = \{(\boldsymbol{x}; y) \in \mathbb{R}^{n+1} \,|\, \boldsymbol{x} = \bar{\boldsymbol{x}},\, y \in [l_{\bar{x}}, u_{\bar{x}}]\}$. Hyper-rectangles $R_y^{\bar{x}}$ satisfy the conditions of Lemma 1, and hence admits a rectangular decomposition w.r.t. the index of variable y which is $n+1$. The result follows from Corollary 1. $\square$

Proof of Theorem 2. First, we show that the Algorithm 1 terminates after a finite number of iterations. To this end, we argue that each loop in the algorithm is repeated for a finite number of iterations. The outer *while* loop is executed for each member of the partial assignment set $\hat{\mathcal{X}}$. These partial assignments are generated based on longest r-u paths associated with nodes u in the exact cut set of relaxed DDs $\overline{\mathcal{D}}$, which contain a finite number of paths by definition; see line 22 of the algorithm. Each resulting partial assignment $\hat{\boldsymbol{x}}$ is different due to the structure of DDs that do not admit multiple paths with similar encoding values. As a result, an upper bound for the number of partial assignments that can be included in $\hat{\mathcal{X}}$ is all possible partial assignments of the solution set of $\mathcal{M}$ for $\boldsymbol{x}$ variables. Because the discrete set $\mathcal{P}$ in the description of $\mathcal{M}$ is bounded by assumption, we conclude that $\hat{\mathcal{X}}$ contains a finite number of elements. For the *repeat* loop in lines 6–10, the goal is to obtain the optimal value of the solution set represented by $\underline{\mathcal{D}}$ subject to the constraints of the subproblem $\mathcal{S}(.)$. It follows from the property of BD method that the resulting optimality and feasibility cuts correspond to the extreme points and extreme rays of the dual of the subproblem, which is independent of the choice of the fixed point $\underline{\boldsymbol{x}}$. Since the feasible region of this dual problem is a polyhedron, the set of its extreme points and rays are finite, which yield a finite set of optimality and feasibility cuts that can be added through the execution of this loop. It is also known in the BD literature (Conforti et al. (2014a)) that the same optimality/feasibility cuts cannot be generated repeatedly through different iterations, since the added cuts remain in the description of the master DD due to refinement. A similar argument can be used to show that the number of iterations of the *repeat* loop in lines 18–21 is finite, thereby yielding the result.

Next, we show that the outputs $(\boldsymbol{x}^*, z^*)$ and w^* of Algorithm 1 give an optimal solution and optimal value of $\mathcal{H}$, respectively. To this end, we first prove that $(\boldsymbol{x}^*, z^*)$ is a feasible solution to $\mathcal{H}$ and w^* is a lower bound for its optimal value. This solution is updated at line 12 of Algorithm 1 as the point encoding a longest r-t path of $\underline{\mathcal{D}}$, after being refined with respect to optimality and feasibility cuts generated from subproblems $\mathcal{S}(.)$. We write that $(\boldsymbol{x}^*, z^*) \in \text{Sol}(\underline{\mathcal{D}}) \subseteq \mathcal{M}^C(\hat{\boldsymbol{x}}) \subseteq \mathcal{M}$, where the second inclusion holds because $\underline{\mathcal{D}}$ is a restricted DD associated with $\mathcal{M}^C(\hat{\boldsymbol{x}})$ for some C and $\hat{\boldsymbol{x}}$, and the third inclusion follows from the fact that a feasible solution to $\mathcal{M}^C(\hat{\boldsymbol{x}})$ is feasible to $\mathcal{M}$ by definition. Further, the BD structure implies that the point encoding a longest path obtained at the termination of the loop in line 6–10 satisfies the constraints of the subproblem $\mathcal{S}(.)$. As a result, $(\boldsymbol{x}^*, z^*)$ satisfies the constraints of both the master and the subproblem, hence being feasible to $\mathcal{H}$. Using a similar argument, we obtain that w^* is a lower bound for the optimal value of $\mathcal{M}$ subject to constraints in $\mathcal{S}(.)$, since w^* is the length of the longest path in the restricted DD representing a restriction of $\mathcal{M}$, after valid optimality and feasibility cuts are added based on the subproblem. Now, we show that $(\boldsymbol{x}^*, z^*)$ is indeed an optimal solution to $\mathcal{H}$. Assume by contradiction that there exists an optimal solution $(\tilde{\boldsymbol{x}}, \tilde{z})$ of $\mathcal{H}$ with optimal value $\tilde{w}$ such that $\tilde{w} > w^*$. There are three cases. (i) Assume that $\tilde{\boldsymbol{x}}$ is added as a partial assignment to set $\hat{\mathcal{X}}$ at some iteration of the algorithm. For the while loop where $\tilde{\boldsymbol{x}}$ is selected at line 3, all $\boldsymbol{x}$ variables are fixed for the restricted DD $\underline{\mathcal{D}}$. Therefore, $\tilde{\boldsymbol{x}}$ is used as the input for the subproblem, i.e., we solve $\mathcal{S}(\tilde{\boldsymbol{x}})$, which yields the optimal value $\tilde{z}$ since $(\tilde{\boldsymbol{x}}, \tilde{z})$ is an optimal solution of $\mathcal{H}$. Since the weights on the arcs of the restricted DD $\underline{\mathcal{D}}$ are set as the coefficients of variables in the linear objective function of $\mathcal{H}$, the length of the r-t path of $\underline{\mathcal{D}}$ encoded by $(\tilde{\boldsymbol{x}}, \tilde{z})$ is $\tilde{w}$. If follows from the contradiction assumption that $\tilde{w} > w^*$ in line 11 of Algorithm 1, which leads to updating the optimal solution and optimal value to $(\tilde{\boldsymbol{x}}, \tilde{z})$ and $\tilde{w}$. As a result, $(\boldsymbol{x}^*, z^*)$ and w^* cannot be returned by the algorithm as the output, a contradiction. (ii) Assume that $\tilde{\boldsymbol{x}}$ is not added as a partial assignment to set $\hat{\mathcal{X}}$ because the algorithm terminates before such a partial assignment is reached in line 22. This implies that there must be a partial assignment $\hat{\boldsymbol{x}} \in \hat{\mathcal{X}}$ with $\hat{x}_i = \tilde{x}_i$ for $i = 1, \dots, j-1$ for some $j \in N$ such that the relaxed DD $\overline{\mathcal{D}}$ associated with $\mathcal{M}^C(\hat{\boldsymbol{x}})$ for some C is pruned without reaching line 22. The only possibility for this event is that the length $\bar{w}$ of the longest r-t path of $\overline{\mathcal{D}}$ must satisfy $\bar{w} \leq w^*$, violating the condition in line 17. However, because of the facts that $\overline{\mathcal{D}}$ is a relaxed DD associated with $\mathcal{M}^C(\hat{\boldsymbol{x}})$, and that $(\tilde{\boldsymbol{x}}, \tilde{z})$ must be a feasible solution to $\mathcal{M}^C(\hat{\boldsymbol{x}})$, we conclude that $\bar{w} \geq \tilde{w}$. Combining this inequality with that of the line above, we obtain $\tilde{w} \leq w^*$, a contradiction to the initial assumption on the optimal value of the problem. (iii) Assume that $\tilde{\boldsymbol{x}}$ is not added as a partial assignment to set $\hat{\mathcal{X}}$ because the longest r-u path chosen in line 22 deviates from $\tilde{\boldsymbol{x}}$ for some node u of the the path associated with $\tilde{\boldsymbol{x}}$. This implies that there must be a partial assignment $\hat{\boldsymbol{x}} \in \hat{\mathcal{X}}$ with $\hat{x}_i = \tilde{x}_i$ for $i = 1, \dots, j-1$ for some $j \in N$ such that the relaxed DD $\overline{\mathcal{D}}$ associated with $\mathcal{M}^C(\hat{\boldsymbol{x}})$ for some C has an exact cut set that contains node u in some layer $k \geq j$. Let $\dot{\boldsymbol{x}}$ be the solution encoding the longest r-u path of $\overline{\mathcal{D}}$ that is chosen in line 22 in place of $\tilde{\boldsymbol{x}}$. It follows from definition of exact cut set that an extension of $\dot{\boldsymbol{x}}$ with components $\dot{x}_i = \tilde{x}_i$ for $i = k, \dots, n$ and $\dot{z} = \tilde{z}$ is a feasible solution to $\mathcal{H}$. However, the above assumption implies that the length $\dot{w}$ of the path encoded by $(\dot{\boldsymbol{x}}, \dot{z})$ is no less than $\tilde{w}$. If $\dot{w} > \tilde{w}$, then we have reached a contradiction to the assumption that $\tilde{w}$ is the optimal value of $\mathcal{H}$. If $\dot{w} = \tilde{w}$, we may repeat the contradiction arguments for the new point $(\dot{\boldsymbol{x}}, \dot{z})$ until we exhaust all possible replacements for the assumed optimal solution. The last such solution must either fall in case (i) or case (ii) above, reaching contradiction. $\square$

Proof of Theorem 3. First, we give a useful observation from Algorithm 2. It follows from this algorithm that state values (s_u^+, s_u^-) at each node $u \in \mathcal{U}_j$ records the number of time periods passed since the last start-up and shut-down of the unit at time j, as these values reset to 1 whenever there is change in the unit status over two consecutive periods, and are incremented by one if the unit status remains the same. These state values imply the status of the unit at the beginning of each period, i.e., the unit is down when $s_u^+ \geq s_u^-$, and it is up otherwise.

For the direct implication, assume that $(\dot{\boldsymbol{x}}, \dot{z})$ encodes an r-t path of length $\dot{c}$ in the equivalence class formed by $\mathcal{D}$. We construct an extended solution $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}, \dot{\bar{\boldsymbol{y}}}, \dot{\boldsymbol{q}}, \dot{z})$ of (2) with objective value $\dot{c}$ as follows. First, we show that $\dot{x}_j \in \{0,1\}$ for all $j \in \mathcal{T}$ and $\dot{z} \in [-\Gamma, \Gamma]$, thereby satisfying domain constraints in (2). The definition of equivalence class implies that, for each $j \in \mathcal{T}$, there exists a node pair $(u,v) \in \mathcal{A}_j \times \mathcal{A}_{j+1}$ such that $\dot{x}_j \in [l^{\min}_{(u,v)}, l^{\max}_{(u,v)}]$, i.e., the variable value belongs to the interval defined by the min and max label values of the arcs connecting nodes u and v; see Section 2.3. It follows from the construction of $\mathcal{D}$ in Algorithm 2 that two consecutive nodes in layers $1, \ldots, T$ cannot be connected with multiple arcs with different label values, since different arc labels lead to different state values for the head node. As a result, we must have $l^{\min}_{(u,v)} = l^{\max}_{(u,v)} \in \{0,1\}$. The argument for $\dot{z} \in [-\Gamma, \Gamma]$ follows directly from the construction and label values assigned to the arcs at the last layer of $\mathcal{D}$. To construct the extended solution, for each $j \in \mathcal{T}$, define $\dot{y}_j = 1$ if $\dot{x}_{j-1} = 0$ and $\dot{x}_j = 1$, and $\dot{y}_j = 0$ otherwise. Similarly, define $\dot{\bar{y}}_j = 1$ if $\dot{x}_{j-1} = 1$ and $\dot{x}_j = 0$, and $\dot{\bar{y}}_j = 0$ otherwise. These definitions guarantee the satisfaction of (1c). Further, let $u \in \mathcal{U}_j$ be the node at layer j of the path encoding $(\dot{\boldsymbol{x}}, \dot{z})$, and define $\dot{q}_j = K_{s_u^-}$ if $\dot{x}_j = 1$ and $\dot{x}_{j-1} = 0$, and $\dot{q}_j = 0$ otherwise. These value assignments satisfy (1b) because this constraint implies that when a unit changes status from down to up, i.e., $x_j = 1$ and $x_{j-1} = 0$, then $q_j \geq \max_{k=1,\ldots,s_u^-} K_k$, as s_u^- represents the number of periods that the unit has been down consecutively before going up. Since the start-up cost function is assumed to be logarithmic, we have that $\max_{k=1,\ldots,s_u^-} K_k = K_{s_u^-}$. Further, since $\dot{q}_j$ takes the maximum value at equality, it yields a non-redundant solution. It remains to show that the constructed point satisfies constraints (1d) and (1e). Assume by contradiction that there exists $j^* \in \mathcal{T}$ for which constraint (1d) is violated by $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}, \dot{\bar{\boldsymbol{y}}}, \dot{\boldsymbol{q}}, \dot{z})$. Note that (1d) models two requirements at time j^*: (i) if the unit is down, then it could not have started up in the last L time periods; and (ii) if the unit is up, then it could not have started up more than once in the last L time periods. For the contradiction, assume first that condition (i) is violated, i.e., $\dot{x}_{j^*} = 0$ and there exists $\bar{j} \in \{j^* - L + 1, \ldots, j^*\}$ such that $\dot{y}_{\bar{j}} = 1$. It follows from construction that $\dot{x}_{\bar{j}} = 1$ and $\dot{x}_{\bar{j}-1} = 0$. Let $(u,v) \in \mathcal{U}_{\bar{j}} \times \mathcal{U}_{\bar{j}+1}$ be the nodes at layers $\bar{j}$ and $\bar{j}+1$ of the r-t path encoding $(\dot{\boldsymbol{x}}, \dot{z})$. We have $s_u^+ \geq s_u^-$ because of the earlier argument on the state values. Since the arc connecting u to v on the r-t path is a 1-arc ($\dot{x}_{\bar{j}} = 1$), we have $s_v^+ = 1$; see line 6 of Algorithm 2. It follows from the algorithm steps that the only way for the unit to be able to shut down after $\bar{j}$ is to satisfy the condition of line 9, i.e., $s_h^+ \geq L$ for some node h in layers $\bar{j}+L,\dots,T$. Since $\bar{j} \geq j^* - L + 1$ by definition, we obtain that $j^* < \bar{j} + L$, and hence $\dot{x}_{j^*}$ cannot be equal to zero, a contradiction. Next, assume that condition (ii) above is violated for the contradiction assumption, i.e., the unit starts up more than once in periods $j^* - L + 1, \dots, j^*$. It is easy to verify that there exists a layer $\tilde{j} \in \{j^* - L + 1, \dots, j^*\}$ for which condition (i) is violated. Therefore, an argument similar to that of condition (i) yields the desired contradiction. The contradiction for violating constraint (1e) is obtained similarly due to the symmetry in the problem structure. We conclude that $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}, \dot{\bar{\boldsymbol{y}}}, \dot{\boldsymbol{q}}, \dot{z})$ is feasible to (2). We next show that the length of the r-t path encoding $(\dot{\boldsymbol{x}}, \dot{z})$ matches the objective value of $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}, \dot{\bar{\boldsymbol{y}}}, \dot{\boldsymbol{q}}, \dot{z})$. The proof follows from considering the contribution of three terms in the objective function of (2). First, the contribution of each variable assignment $\dot{x}_j = 1$ to the objective function is c_f, which is captured in the weight of the associated 1-arcs of the r-t path through Algorithm 2 in lines 6 and 8. Second, the contribution of the start-up status of the unit to the objective function is $\dot{q}_j = K^1_{s_u^-}$ where u is the node at layer j of the path encoding $(\dot{\boldsymbol{x}}, \dot{z})$, which is captured in line 6 of Algorithm 2. Third, the contribution of $\dot{z}$ in the objective function is directly considered in the arc weight in the last layer of the equivalence class of $\mathcal{D}$.

For the reverse implication, assume that $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}, \dot{\bar{\boldsymbol{y}}}, \dot{\boldsymbol{q}}, \dot{z})$ is a non-redundant solution of (2) with objective value $\dot{c}$. We show that the projected point $(\dot{\boldsymbol{x}}, \dot{z})$ encodes an r-t path of length $\dot{c}$ in the equivalence class formed by $\mathcal{D}$. Assume by contradiction that no such path exists in the equivalence class of $\mathcal{D}$. Therefore, there exists an r-t path $\mathtt{P}$ of $\mathcal{D}$ with associated point $(\bar{\boldsymbol{x}}, \bar{z})$ and a layer number $j^* \in \mathcal{T} \setminus \{T\}$ such that $\bar{x}_j = \dot{x}_j$ for $j \in \{1, \dots, j^*\}$ and the node u at layer $j^* + 1$ of $\mathtt{P}$ does not have an outgoing arc with label value $\dot{x}_{j^*+1}$. There are four cases. For the first case, assume that $\dot{x}_{j^*} = 0$ and $\dot{x}_{j^*+1} = 0$. It follows from the state value definitions that $s_u^+ \geq s_u^-$ as $\dot{x}_{j^*} = \bar{x}_{j^*} = 0$. Line 4 of Algorithm 2 implies that u has an outgoing arc with label value $\dot{x}_{j^*+1} = 0$, a contradiction. For the second case, assume that $\dot{x}_{j^*} = 1$ and $\dot{x}_{j^*+1} = 0$. Because of constraints (1c) and (1d), we must have $\dot{x}_j = \bar{x}_j = 1$ for $j = j^* - L + 1, \dots, j^*$. Let v be the node at layer $j^* - L + 1$ of $\mathtt{P}$. Since $\bar{x}_j = 1$ for $j = j^* - L + 1, \dots, j^*$, it follows from line 8 of Algorithm 2 that $s_u^+ = s_v^+ + L - 1 \geq L$ as $s_v^+ \geq 1$. As a result, the condition of line 9 of Algorithm 2 is satisfied at node u, which leads to creating a 0-arc as an outgoing arc of u, a contradiction. For the third case, where $\dot{x}_{j^*} = 1$ and $\dot{x}_{j^*+1} = 1$, the contradiction is obtained similarly to the first case due to the symmetry of the problem. For the fourth case, where $\dot{x}_{j^*} = 0$ and $\dot{x}_{j^*+1} = 1$, the contradiction is achieved similarly to the second case due to the symmetry of the problem. Finally, since $\dot{z} \in [-\Gamma, \Gamma]$, it must belong to the equivalence class of $\mathcal{D}$ as $-\Gamma$ and Γ are used as label values of the pair of arcs connecting every node of layer $T + 1$ to the terminal node in $\mathcal{D}$. We conclude that $(\dot{\boldsymbol{x}}, \dot{z})$ encodes an r-t path of length $\dot{c}$ in the equivalence class formed by $\mathcal{D}$. To show that the objective value $\dot{c}$ is equal to the length of the r-t path encoding $(\dot{\boldsymbol{x}}, \dot{z})$, we note that $\dot{q}^1_j = K^1_{s_h^-}$ where h is the node at layer j of the r-t path, since otherwise $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}, \dot{\bar{\boldsymbol{y}}}, \dot{\boldsymbol{q}}, \dot{z})$ would be a redundant solution of (2). The rest of the proof follows from an argument similar to that of the direct implication case given above. $\square$

Proof of Proposition 3. We need to show that for each path $\bar{\mathsf{P}}$ of $\bar{\mathcal{D}}$ with encoding point $(\bar{\boldsymbol{x}}, \bar{z})$, there exists a path $\tilde{\mathsf{P}}$ of $\tilde{\mathcal{D}}$ with encoding point $(\tilde{\boldsymbol{x}}, \tilde{z})$ such that $\bar{\boldsymbol{x}} = \tilde{\boldsymbol{x}}$, $\bar{z} = \tilde{z}$, and $w(\bar{\mathsf{P}}) \geq w(\tilde{\mathsf{P}})$, where $w(.)$ represent the length of the path. Define $\bar{\mathsf{P}}^k$ to be the path composed of the first k arcs of $\bar{\mathsf{P}}$ for $k \in \mathcal{T}$. We prove the result by induction on the size of k, i.e., we show that for each $\bar{\mathsf{P}}^k$ with encoding point $\bar{\boldsymbol{x}}^k$, there exists a path $\tilde{\mathsf{P}}^k$ of $\tilde{\mathcal{D}}$ with encoding point $\tilde{\boldsymbol{x}}^k$ such that $\bar{\boldsymbol{x}}^k = \tilde{\boldsymbol{x}}^k$ and $w(\bar{\mathsf{P}}^k) \geq w(\tilde{\mathsf{P}}^k)$. Note that it suffices to show the above result for k up to T, since each node at node layer $T+1$ is connected to the terminal node by two arcs with labels $-\Gamma$ and Γ in both $\bar{\mathcal{D}}$ and $\tilde{\mathcal{D}}$, and therefore a matching arc can always be found for the desired path. For the base of induction, i.e., $k = 1$, it follows from Algorithm 2 that $\bar{\mathcal{D}}$ has two arcs with labels 0 and 1 going out of the root node. These two arcs remain in $\tilde{\mathcal{D}}$ with the same weights according to Definition 2. For the induction hypothesis, assume that, for some $k \geq 2$, there exists a path $\tilde{\mathsf{P}}^{k-1}$ of $\tilde{\mathcal{D}}$ with encoding point $\tilde{\boldsymbol{x}}^{k-1}$ such that $\bar{\boldsymbol{x}}^{k-1} = \tilde{\boldsymbol{x}}^{k-1}$ and $w(\bar{\mathsf{P}}^{k-1}) \geq w(\tilde{\mathsf{P}}^{k-1})$. We show that there exists a path $\tilde{\mathsf{P}}^k$ of $\tilde{\mathcal{D}}$ with encoding point $\tilde{\boldsymbol{x}}^k$ such that $\bar{\boldsymbol{x}}^k = \tilde{\boldsymbol{x}}^k$ and $w(\bar{\mathsf{P}}^k) \geq w(\tilde{\mathsf{P}}^k)$. There are four cases. (i) Assume that $\bar{x}^k_{k-1} = 0$ and $\bar{x}^k_k = 0$. Note that $\bar{x}^k_{k-1} = \bar{x}^{k-1}_{k-1} = 0$ as they are defined over the same path $\bar{\mathsf{P}}$. It follows from the induction hypothesis that $\bar{x}^{k-1}_{k-1} = \tilde{x}^{k-1}_{k-1} = 0$. Let u be the node at node layer k of the path $\tilde{\mathsf{P}}^{k-1}$. We consider two cases for this node. For the first case, assume that u is not a merged node. As a result, it has been created directly from the modified algorithm of Definition 2, which implies that $s^+_u \geq s^-_u$. Following line 4 of Algorithm 2, there is a 0-arc going out of u in $\tilde{\mathcal{D}}$. Adding this arc to $\tilde{\mathsf{P}}^{k-1}$ yields the desired path $\tilde{\mathsf{P}}^k$, as the contribution to the objective function is 0 for this variable assignment in both paths $\bar{\mathsf{P}}^k$ and $\tilde{\mathsf{P}}^k$. For the second case, assume that u is a merged node. Let $\{v_1, \dots, v_r\}$ be the set of nodes that have been merged into u using the merging operation of Definition 3. Since u has an incoming arc with label 0, it means that one of the nodes v_i in the above set must have an incoming arc with label 0. The modified algorithm implies that $s^+_{v_i} \geq s^-_{v_i}$. The merging condition, then, dictates that $s^+_{v_i} \geq s^-_{v_i}$ for all $i \in \{1, \dots, r\}$. It follows from the merging equations that $s^+_u = \max_i s^+_{v_i} \geq \max_i s^-_{v_i} = s^-_u$. Therefore, an argument similar to that of the first case yields the desired result. (ii) Assume that $\bar{x}^k_{k-1} = 0$ and $\bar{x}^k_k = 1$. Let h be the node at node layer k of $\bar{\mathsf{P}}^k$. It follows from line 5 of Algorithm 2 that $s^-_h \geq \ell$ and that the arc weight at layer k is $c_f + K_{s^-_h}$. It is clear from the definition of s^-_h that $\bar{x}^k_{k-s^-_h-1} = 1$ and $\bar{x}^k_{k-s^-_h-1+i} = 0$ for all $i = 1, \dots, s^-_h$. It follows from the induction hypothesis that $\tilde{x}^{k-1}_{k-s^-_h-1} = 1$ and $\tilde{x}^{k-1}_{k-s^-_h-1+i} = 0$ for all $i = 1, \dots, s^-_h$. Let u and v be the nodes at node layers $k - s^-_h + 1$ and k of $\tilde{\mathsf{P}}^{k-1}$, respectively. We consider two cases for node u. For the first case, assume that u is not a merged node. As a result, it has been created directly from the modified algorithm of Definition 2, which implies that

$s^-_u = s^=_u = 1$. For the second case, assume that u is a merged node. Let $\{u_1, \dots, u_r\}$ be the set of nodes that have been merged into u using the merging operation of Definition 3. Since u has an incoming arc with label 0 right after an arc with label 1, it implies that one of the nodes u_i in the above set must have $s^-_{u_i} = s^=_{u_i} = 1$. It follows from the modified algorithm that $s^-_u \geq 1$ and $s^=_u \leq 1$, which holds for the first case as well. Using an argument similar to that given above for the nodes at layers $k - s^-_h + 1$ to k of $\tilde{\mathsf{P}}^{k-1}$, we conclude that $s^-_v \geq s^-_u + s^-_h - 1 \geq s^-_h$ and $s^=_v \leq s^=_u + s^-_h - 1 \leq s^=_h$, where the first inequalities follow from the state definitions through merging operation applied on successive node layers, and the second inequalities hold because of the relations above on state values at node u. As a result, the condition in line 5 of Algorithm 2 is satisfied at node v as $s^-_v \geq s^-_h \geq \ell$, meaning that it has an outgoing arc with label 1 and weight $c_f + K_{s^=_v} \leq c_f + K_{s^=_h}$. Adding this arc to $\tilde{\mathsf{P}}^{k-1}$ yields the desired path $\tilde{\mathsf{P}}^k$. (iii) Assume that $\bar{x}^k_{k-1} = 1$ and $\bar{x}^k_k = 1$. The result is obtained by using an argument similar to that of case (i) due to symmetry. (iv) Assume that $\bar{x}^k_{k-1} = 1$ and $\bar{x}^k_k = 0$. An argument similar to that of case (ii) proves the result due to symmetry. □

Proof of Proposition 4. We prove the equivalence by showing that for each feasible variable assignment, both sets of constraints impose the same restrictions. There are four cases for each $i \in N$ and $j \in \mathcal{T}$. (i) Assume that $x^i_{j-1} = x^i_j = 0$. It follows from constraints (1c) and (1d) that $y^i_j = \bar{y}^i_j = 0$. In this case, the right-hand-side (RHS) of both inequalities (1f) and (4a), as well as (1g) and (4b) reduce to zero. (ii) Assume that $x^i_{j-1} = 0$ and $x^i_j = 1$. It follows from constraint (1c) that $y^i_j = 1$ and $\bar{y}^i_j = 0$. In this case, the RHS of both constraints (1f) and (4a) reduce to SU^i. Further, inequality (1h) implies that $p^i_{j-1} = 0$ as $x^i_{j-1} = 0$. As a result, both constraints (1g) and (4b) become redundant in the description of $\mathcal{E}$ and $\mathcal{G}$, since their LHS is a non-positive quantity $-p^i_j \leq 0$ and their RHS are non-negative values as $RD^i \geq 0$ and $RD^i - SD^i \geq 0$ by assumption. (iii) Assume that $x^i_{j-1} = 1$ and $x^i_j = 0$. It follows from constraint (1c) that $y^i_j = 0$ and $\bar{y}^i_j = 1$. In this case, the RHS of both constraints (1g) and (4b) reduce to SD^i. Further, inequality (1h) implies that $p^i_j = 0$ as $x^i_j = 0$. As a result, both constraints (1f) and (4a) become redundant in the description of $\mathcal{E}$ and $\mathcal{G}$, since their LHS is a non-positive quantity $-p^i_{j-1} \leq 0$ and their RHS are non-negative values as $RU^i \geq 0$ and $RU^i - SU^i \geq 0$ by assumption. (iv) Assume that $x^i_{j-1} = x^i_j = 1$. It follows from constraints (1c) and (1e) that $y^i_j = \bar{y}^i_j = 0$. In this case, the RHS of both (1f) and (4a) reduce to RU^i, and the RHS of both (1g) and (4b) reduce to RD^i. □

Appendix B: Comparison of DD-BD Method with the Literature

In the literature, DDs have been used in conjunction with BD framework in different contexts. In this section, we give a detailed comparison between DD-BD approach of Section 3 and those used in the literature. In particular, we show that our framework generalizes other existing approaches, while addressing their shortcomings.

van der Linden (2017) proposes a BD framework for linear mixed integer programs that uses a similar approach to ours in defining the master problem over integer variables, and solving the subproblems for continuous variables. A DD is constructed to represent the solution set of the master problem, and is successively refined through cuts obtained from the subproblems. There is, however, a fundamental difference between our approach and that of van der Linden (2017) as they use separation for the feasibility cuts only since their proposed DD cannot contain any continuous variable. To incorporate the optimality cuts, the authors propose a so-called "cost-tuple" approach where, for each optimality cut, a cost parameter is calculated at each node of the DD to record the contribution of the variable assignments in relation to that cut. Such an approach requires the underlying DD to be *non-reduced*, limiting its practicality. In contrast, our proposed DD-BD approach provides a unified framework that treats both feasibility and optimality cuts similarly through separation as the continuous variable is directly incorporated in the DD layers. To elaborate, we first give a brief review of the cost-tuple approach used in van der Linden (2017) to handle optimality cuts.

The cost-tuple approach is given in Algorithm 3 where, without loss of generality, the master problem of a MIP is defined as $\max\{z \,|\, \boldsymbol{x} \in \mathcal{P}\}$ with a bounded $\mathcal{P} \subseteq \mathbb{Z}^n$. Consider $\mathcal{D} = (\mathcal{U}, \mathcal{A}, l(\cdot))$ to be the DD representing $\mathcal{P}$, with a property that each node (except the terminal) has a unique incoming arc, and the terminal receives a unique arc from each node of the previous layer. Let inequalities of the form $z \leq \boldsymbol{\alpha}^j \boldsymbol{x} + \alpha_0^j$, for $j \in J$, represent the set of added optimality cuts at the current iteration of the BD algorithm. For each node $u \in \mathcal{U}$, parameter $r^j(u)$ records the reward (cost, for a minimization variant) contribution of variable assignments at that node to the right-hand-side value of cut $z \leq \boldsymbol{\alpha}^j \boldsymbol{x} + \alpha_0^j$ for $j \in J$. In this algorithm, the accumulated reward $r(u)$ captures the contribution amount at the intersection of optimality cuts for each unique value assignment of variables, while the final reward r^* computes the optimal value of $\max\{z \,|\, \boldsymbol{x} \in \mathcal{P}, z \leq \boldsymbol{\alpha}^j \boldsymbol{x} + \alpha_0^j, \forall j \in J\}$.

The main shortcoming of Algorithm 3 is due to the assumption that the DD nodes have a unique incoming arcs. This requirement leads to an exponential growth of the DD size to distinguish between each r-t path. As a consequence, it is mentioned in van der Linden (2017) that the cost-tuple algorithm is not applicable to reduced DDs. Reduction in DDs is referred to the operation of merging nodes that share a similar subtree; hence it is regarded as a crucial component in building efficient DDs for practical applications and keeping them within a width limit. The DD-BD approach proposed in the present paper, on the other hand, can be applied to reduced DDs, and the separation of merged nodes, if needed, will be performed naturally through the separation operation. We illustrate this difference in the following example.

Algorithm 3: The cost-tuple approach of van der Linden (2017) to handle optimality cuts

```
Data: a DD $\mathcal{D} = (\mathcal{U}, \mathcal{A}, l(.))$ and a set of optimality cuts $z \leq \boldsymbol{\alpha}^j \boldsymbol{x} + \alpha_0^j$, for $j \in J$
1 initialize the root node reward as $r^j(s) = \alpha_0^j$ for $j \in J$
2 forall cuts $j \in J$, node layers $i \in N \setminus \{n\}$, and node pairs $(u, v) \in \mathcal{U}_i \times \mathcal{U}_{i+1}$ do
3     compute $r^j(v) = r^j(u) + \alpha_i^j l_a$ where $a \in \mathcal{A}_i$ is the unique arc connecting $u$ to $v$
4 forall nodes $u \in \mathcal{U}_n$ do
5     compute accumulated reward $r(u) = \min_{j \in J}\{r^j(u) + \alpha_n^j l_a\}$ where $a \in \mathcal{A}_n$ is the unique
       arc connecting $u$ to the terminal $t$
6 compute the final reward $r^* = \max_{u \in \mathcal{U}_n} r_u$
```

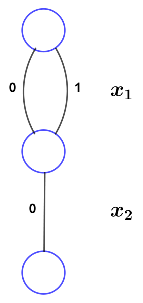
Example 5. Consider the final iteration of a BD algorithm with a master problem $\max\{z \mid \boldsymbol{x} \in \mathcal{P}\}$ where $\mathcal{P} \subseteq \mathbb{Z}^n$ is represented by DD $\mathcal{D}_1$ of Figure 8a. Assume that the following two optimality cuts are added

$$z \leq 3x_1 + 2x_2 \tag{9a}$$

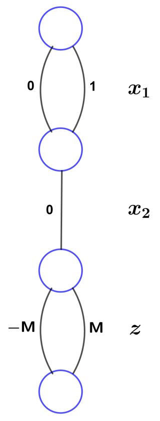
$$z \leq -3x_1 - 5x_2 + 3 \tag{9b}$$

It is clear that both points $(0,0)$ and $(1,0)$ give an optimal solution with optimal value $z^* = 0$. Since $\mathcal{D}_1$ is a reduced DD, the approach of van der Linden (2017) cannot be directly applied to obtain the optimal solution. It is suggested in that work that, when merging occurs, the component-wise maximum of the reward tuple for merging nodes must be selected as the reward tuple of the merged node. This operation guarantees that the final reward value is an upper-bound for the optimal reward value. Applying this operation here, however, creates an infinite loop without converging to a solution, as demonstrated next. We record the reward value for inequalities (9a) and (9b), respectively, in a tuple $(r^1(u), r^2(u))$ for each node u. For the root node, we have the reward tuple $(0,3)$ according to Algorithm 3. For the path including the 1-arc at the first layer, the reward of the end-node is $(3,0)$. Similarly, the path including the 0-arc at the first layer has the end-node reward $(0,3)$. The component-wise maximum yields the reward of $(3,3)$ for the merged node in the middle. The reward at the terminal node is $(3,3)$, giving the optimal value $\bar{z} = 3$. While $\bar{z}$ is an upper-bound for z^*, it can already be separated by the current optimality cuts (9a) and (9b) for both feasible solutions. As a result, the BD approach cannot refine it any further and is stuck at this iteration.

For the DD-BD approach of Algorithm 1, we represent the master problem as $\mathcal{M} = \max\{z \mid \boldsymbol{x} \in \mathcal{P}, z \in [-M, M]\}$ where valid bounds are imposed on z variable. A restricted DD $\mathcal{D}_2$ can be created for $\mathcal{M}$ using the equivalence class argument of Section 2.3 as given in Figure 8b. Next, we may simply refine the DD through a sequential separation with respect to optimality cuts (9a) and (9b), as shown in Figures 9a and 9b. It is easy to verify that a longest path in the resulting DD gives either point $(0,0)$ or $(1,0)$ with the optimal value $z^* = 0$. Since these solutions give the optimal value of the problem, Algorithm 1 is terminated due to bound pruning conditions in line 17.

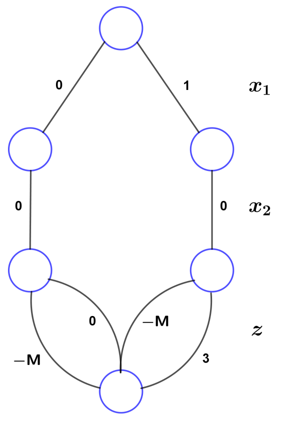


(a) $\mathcal{D}_1$ without continuous variable

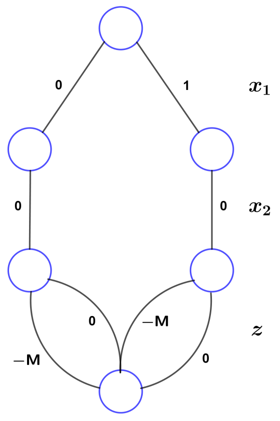


(b) $\mathcal{D}_2$ with continuous variable

Figure 8 DD representation of the master problem of Example 5



(a) Separation wrt (9a)



(b) Separation wrt (9b)

Figure 9 Separation of optimality cuts for the master problem of Example 5

We conclude this section by showing that the DD-BD approach of Algorithm 1 is a generalization of the cost-tuple approach of Algorithm 3, since the latter can be viewed as a special case of the former when the underlying DD is not reduced.

PROPOSITION 5. *Consider the master problem* $\max\{z \,|\, \boldsymbol{x} \in \mathcal{P}\}$ *of a MIP with a bounded* $\mathcal{P} \subseteq \mathbb{Z}^n$*, and let* $\mathcal{D} = (\mathcal{U}, \mathcal{A}, l(\cdot))$ *be the DD representing* $\mathcal{P}$*, with a property that each node (except the terminal) has a unique incoming arc, and the terminal receives a unique arc from each node of the previous layer. Let inequalities of the form* $z \leq \boldsymbol{\alpha}^j \boldsymbol{x} + \alpha^j_0$*, for* $j \in J$*, represent the set of optimality cuts to be added at the current iteration of the BD algorithm. Define* $\bar{\mathcal{D}} = (\bar{\mathcal{U}}, \bar{\mathcal{A}}, \bar{l}(\cdot))$ *to be the DD constructed from* $\mathcal{D}$ *by adding a layer representing* z *variable as the last arc layer. Then, the reward value obtained from Algorithm 3 applied to* $\mathcal{D}$ *is equal to the optimal value obtained from Algorithm 1 when applied to* $\bar{\mathcal{D}}$*.*

Proof. We first describe the relation between $\mathcal{D}$ and $\bar{\mathcal{D}}$. All nodes and arcs from layer one to layer $n-1$ are similar in both DDs, i.e., $\mathcal{A}_i = \bar{\mathcal{A}}_i$ for $i \in N \setminus \{n\}$ and $\mathcal{U}_i = \bar{\mathcal{U}}_i$ for $i \in N$. The difference is that $\bar{\mathcal{D}}$ has an extra layer that represents z variable. This layer is created as follows. For each node $u \in \bar{\mathcal{U}}_n$, we create a node $v \in \bar{\mathcal{U}}_{n+1}$, and connect them with an arc $a \in \bar{\mathcal{A}}_n$ with the same label as the unique arc connecting the replica of node u in $\mathcal{U}_n$ to the terminal node of $\mathcal{D}$. Then, for each node $v \in \bar{\mathcal{U}}_{n+1}$, we create two arcs with labels $-M$ and M in layer $\bar{\mathcal{A}}_{n+1}$ that connect v to the terminal node in $\bar{\mathcal{D}}$. These label values represent some valid lower and upper bounds for variable z. According to Algorithm 1, $\bar{\mathcal{D}}$ is refined with respect to the optimality cuts, and then the longest path is found over the refined DD weighted by the objective function rates. To perform the refinement, the solution set satisfying each optimality cut is modeled by a DD called *threshold diagram*; see Bergman et al. (2016) for an exposure to threshold diagrams. The last layer of such DDs corresponds to variable z and contains two arcs with labels representing a lower and upper bound for z. Since for each node $v \in \bar{\mathcal{U}}_{n+1}$, there exists a unique arc-specified path $\mathrm{P} = (a_1, \dots, a_n) \in \bar{\mathcal{A}}_1 \times \dots \times \bar{\mathcal{A}}_n$ from the root node to v, it follows from the assumption that the threshold diagrams have the same structure as $\bar{\mathcal{D}}$, with the difference that the upper bound label on arcs connecting v to the terminal is computed as $M^j(v) = \sum_{i=1}^n \alpha^j_i l_{a_i} + \alpha^j_0$ for each optimality cut $j \in J$. As a result, refining $\bar{\mathcal{D}}$ with respect to the optimality cuts reduces to intersecting label value intervals associated with arcs at the last layer, which are of the form $[-M, M^j(v)]$ for each node $v \in \bar{\mathcal{U}}_{n+1}$. This intersection yields the lower bound label $-M$, and the upper bound label $M(v) = \min_{j \in J} M^j(v)$ for pair of arcs connected to v in the last layer of the refined DD. Note that $M(v)$ is equal to $r(v)$ as computed in Algorithm 3. Finally, to find the longest path on the refined $\bar{\mathcal{D}}$, all arcs in layers 1–n are assigned zero weight and arcs in the last layer are assigned weight 1 deduced from the objective function of the master problem. Therefore, the length of the longest r-t path in $\bar{\mathcal{D}}$ is computed as $w^* = \max_{v \in \bar{\mathcal{U}}_{n+1}} M(v)$, which is equal to r^* obtained from Algorithm 3. Since $\bar{\mathcal{D}}$ represents an exact DD for the master problem and has been refined with respect to optimality cuts, w^* is returned as the output of Algorithm 1. $\square$