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 on 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 Tahanan et al. (2015) 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-Españo et al. (2013) propose a stronger MIP formulation than those of Carrió 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. Appendix C contains detailed tables for numerical results.

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 $N := \{1, \ldots, n\}$. We refer to a DD by $D = (\mathcal{U}, \mathcal{A}, l(\cdot))$, where $\mathcal{U}$ denotes the set of nodes and $\mathcal{A}$ represents the set of arcs, and function $l: \mathcal{A} \rightarrow \mathbb{R}$ indicates the label of arcs in $\mathcal{A}$. The multi-graph induced by $D$ is composed of $n$ arc layers $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$, and $n + 1$ node layers $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_{n+1}$. 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 over all node layers.

DD $D$ represents a set of points of the form $\mathbf{x} = (x_1, \ldots, x_n)$ 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 $\mathbf{x}$ associated with the labels of the arcs on the path. The collection of points encoded by all paths of $D$ is referred to as the solution set of $D$, which is denoted by $\text{Sol}(D)$.

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 $z^* = \max \{ f(\mathbf{x}) \mid \mathbf{x} \in \mathcal{P} \}$ where $f: \mathbb{R}^n \rightarrow \mathbb{R}$
and \( P \subseteq \mathbb{Z}^n \). To model the above integer program with a DD, we first construct a \textit{weighted} DD, denoted by \([D|w(\cdot)]\), where (i) \( D \) represents an exact DD encoding solutions of \( P \), and (ii) \( w(\cdot): A \rightarrow \mathbb{R} \) is a weight function associated with arcs of \( D \) so that for each \( r\)-\( t \) path \( P = (a_1, \ldots, a_n) \), its weight \( w(P) := \sum_{i=1}^{n} w(a_i) \) is equal to \( f(x^P) \), the objective value of the integral solution corresponding to \( P \). Construction of the weight function is immediate when \( f(x) \) is separable, i.e., \( f(x) = \sum_{i=1}^{n} f_i(x_i) \), as we can simply define \( w(a_i) = f_i(l(a_i)) \) for \( a \in A_i \) 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 \textit{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 \textit{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 \( x_I \) represent coordinates of \( x \in \mathbb{R}^n \) whose index belongs to \( I \). Given a set \( P \subseteq \mathbb{R}^n \), \( \text{conv}(P) \) describes the convex hull of \( P \), \( \text{proj}_{x_I}(P) \) represents the projection of \( P \) onto the space of variables \( x_I \), and \( \text{dim} P \) indicates the dimension of \( \text{conv}(P) \). Given a function \( f(x): \mathbb{R}^n \rightarrow \mathbb{R} \) and a variable subset \( I \subseteq N \), we say that \( f(x) \) is \textit{convex in} \( x_I \) if the restriction of \( f(x) \) 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 \( \text{conv}(P) \).
Theorem 1. Consider a set $\mathcal{P} \subseteq \mathbb{R}^n$, and select $I \subseteq N$. Assume that there exists a finite collection of bounded sets $P^j$ for $j \in J$, where $J$ is an index set, such that

(i) $\dim \text{proj}_{x_i}(P^j) = 0$, for $i \in N \setminus I$ and $j \in J$, i.e., coordinate $x_i$ is fixed.

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

(iii) For each $j \in J$, there exists a finite collection of hyper-rectangles of the form $R^k_j = \prod_{i=1}^n [l^k_i, u^k_i]$ for $k \in K_j$, where $K_j$ is an index set, such that $\text{conv}(P^j) = \text{conv}\left(\bigcup_{k \in K_j} R^k_j\right)$.

Then, $\max\{f(x) | x \in \mathcal{P}\} = \max\{f(x) | x \in \bigcup_{j \in J} \mathcal{X}(P^j)\} = \max\{f(x) | x \in \bigcup_{j \in J} \bigcup_{k \in K_j} R^k_j\}$ for any function $f(x)$ that is convex in $x_1$. $\square$

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

Example 1. Consider a bounded mixed integer set

$$\mathcal{P} = \{(x_1, x_2, x_3) \in [0, 2]^2 \times \{0, 1\} \mid x_1 + x_2 \leq 3; x_3 - x_1 \leq 0; x_1 - x_3 \leq 1\}.$$ 

Set $\mathcal{P}$ is shown in Figure 1. Select $I = \{1, 2\}$. To use the result of Theorem 1, we first introduce sets $P^1 = \{x \in \mathbb{R}^3 \mid 0 \leq x_1 \leq 1; 0 \leq x_2 \leq 2; x_3 = 0\}$, and $P^2 = \{x \in \mathbb{R}^3 \mid 1 \leq x_1 \leq 2; 0 \leq x_2 \leq 2; x_1 + x_2 \leq 3; x_3 = 1\}$. It is easy to verify that conditions (i) and (ii) of Theorem 1 hold for these sets.

Next, define hyper-rectangles $R^1_1 = [0, 1] \times [0, 2] \times \{0\}$ for $P^1$, and $R^1_2 = \{1\} \times [0, 2] \times \{1\}$ and $R^2_2 = \{2\} \times [0, 1] \times \{1\}$ for $P^2$. These hyper-rectangles satisfy condition (iii) of Theorem 1. As a result, $\max\{f(x) | x \in \mathcal{P}\} = \max\{f(x) | x \in \mathcal{X}(P^1) \cup \mathcal{X}(P^2)\} = \max\{f(x) | x \in R^1_1 \cup R^1_2 \cup R^2_2\}$ for any function $f(x)$ 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 $P^j$ through fixing variables not in $I$. Second, each set $P^j$ 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 (Rockafellar 1996), 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.

For instance, consider $P = \{(x_1, x_2) \in [-1, 1] \times \{-1, 0, 1\} | |x_1| \leq |x_2|\}$, and select $I = \{1\}$. Clearly $\mathcal{X}(P) = \{(-1, -1), (-1, 1), (1, -1), (1, 1)\}$. It follows that the extreme point decomposition alone does not provide the desired equivalence as $\max\{-x_2^2 | x \in P\} \neq \max\{-x_2^2 | x \in \mathcal{X}(P)\}$. 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 set $P \subseteq \mathbb{R}^n$, and select $I \subseteq \mathbb{N}$. Assume that there exists a finite set $Q \subseteq \mathbb{R}^n$ such that $\max\{f(x) | x \in P\} = \max\{f(x) | x \in Q\}$ for every function $f(x)$ that is convex in $x_I$. Then, $P$ 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 set $P \subseteq \mathbb{R}^n$, and select $I \subseteq \mathbb{N}$. Assume that there exists a finite collection of hyper-rectangles of the form $R^j = \prod_{k=1}^{n} [l^j_k, u^j_k]$ 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, \( \max \{ f(x) | x \in \mathcal{P} \} = \max \{ f(x) | x \in \bigcup_{j \in J} \mathcal{X}(R_i^j) \} = \max \{ f(x) | x \in \bigcup_{j \in J} R_i^j \} \) for any function \( f(x) \) that is convex in \( x \).

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 \( R_i^j \) 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 \( a \in A \) 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.

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

**Proposition 2.** Consider a DD \( D = (U, A, l(\cdot)) \), and select \( I \subseteq N \).

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

(ii) Let \( \hat{D} \) be a virtual DD constructed from \( D \) with a difference that, for any node pair \((u, v)\) \in \( U_i \times U_{i+1} \) 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, \( \max \{ f(x) | x \in \text{Sol}(D) \} = \max \{ f(x) | x \in \text{Sol}(\hat{D}) \} = \max \{ f(x) | x \in \text{Sol}(\tilde{D}) \} \) for any function \( f(x) \) that is convex in \( x_I \). □

Proposition 2 defines an equivalence class for DDDs. 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 DDDs into regular DDDs. This property can also be useful from a computational perspective as the size of the DDDs 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 DDDs community concerns the characteristics of optimization problems that can be represented through a DD formulation.

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

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 DDDs 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 \( \mathcal{P} \subseteq \mathbb{R}^n \), and select \( I \subseteq \mathbb{N} \). 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 dis-
crete 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 bounded set and define the mixed integer set \( \mathcal{P} = \{ (x; y) \in \mathcal{Q} \mid x \in \mathbb{Z}^n \} \). 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. 2014). 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 exis-
tence 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.
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 \( \max\{f(x) | x \in \mathcal{P}\} = \max\{f(x) | x \in \text{Sol}(\mathcal{D})\} \) for every convex function \( f(x) \). Define \( P_1^I = \mathcal{P} \) with \( I = \{1, 2\} \). An immediate rectangular decomposition is obtained through rectangles \( R_1^I \) 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 \( R_1^I = [0,1] \times [0,2] \) and \( R_2^I = \{2\} \times [0,1] \). 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.
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 \( \mathcal{H} = \max \{ ax + by \mid Ax + By \leq c, x \in \mathbb{Z}^n \} \) where \( a \) and \( b \) are row vectors. We use BD to define the master problem \( \mathcal{M} = \max \{ ax + z \mid (x, z) \in \mathcal{P} \times [l_0, u_0] \} \) where \( \mathcal{P} \subseteq \mathbb{Z}^n \) is the projection of the feasible region of \( \mathcal{H} \) onto \( x \)-space, and the bounds on \( z \) are induced from the boundedness of \( \mathcal{H} \). Subproblems are defined as \( S(\bar{x}) = \max \{ by \mid By \leq c - A\bar{x} \} \) for a given \( \bar{x} \). At each iteration of the algorithm, the output of the subproblems is either a feasibility cut of the form \( \alpha x \leq \alpha_0 \) or an optimality cut of the form \( z + \alpha x \leq \alpha_0 \), where \( \alpha \) 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 \( x \) 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; 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 $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_{|\hat{x}|})$ to be a partial assignment of size $|\hat{x}|$ to variables $x_1, \ldots, x_{|\hat{x}|}$. Define $\mathcal{M}^C(\hat{x}) = \max\{ax + z | (x, z) \in \mathcal{P} \times [l_0, u_0] \cap \mathcal{F}^C, x_i = \hat{x}_i, \forall i = 1, \ldots, |\hat{x}|\}$ to be the restriction of the master problem $\mathcal{M}$ after adding the cuts in $C$ and fixing the partial assignment $\hat{x}$. 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 $\mathcal{M}^C(\hat{x})$ by fixing the path associated with the partial assignment $\hat{x}$ and refining the DD with respect to the cuts in $C$. Given a DD $\mathcal{D}$ with $n$ layers, we denote by $\text{EXACT}(\mathcal{D})$ 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.

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

Data: $\mathcal{H}, \mathcal{M}, S(\vec{x})$, and oracles to build relaxed and restricted DDs for $\mathcal{M}$

Result: An optimal solution $(\vec{x}^*, z^*)$ and optimal value $w^*$ of $\mathcal{H}$

1. initialize set of partial assignments $\hat{X} := \{\emptyset\}$, set of cuts $C := \emptyset$, and lower bound $w^* := -\infty$

2. while $\hat{X} \neq \emptyset$ do

3. pick $\hat{x} \in \hat{X}$ and update $\hat{X} \leftarrow \hat{X} \setminus \{\hat{x}\}$

4. create a restricted DD $\mathcal{D}$ associated with $\mathcal{M}^C(\hat{x})$

5. if $\mathcal{D} \neq \emptyset$ then

6. repeat

7. find a longest $r$-path in $\mathcal{D}$ with encoding point $(\vec{x}, z)$ and length $w$

8. solve $S(\vec{x})$ to obtain feasibility/optimality cuts $C$ and optimal value $\rho$

9. update $C \leftarrow C \cup C$, and refine $\mathcal{D}$ w.r.t $C$

10. until $z = \rho$

11. if $w > w^*$ then

12. update $w^* \leftarrow w$ and $(\vec{x}^*, z^*) \leftarrow (\vec{x}, z)$

13. else

14. Go to line 2

15. create a relaxed DD $\mathcal{D}$ associated with $\mathcal{M}^C(\hat{x})$

16. find a longest $r$-path in $\mathcal{D}$ with encoding point $(\vec{x}, z)$ and length $w$

17. if $w > w^*$ then

18. repeat

19. solve $S(\vec{x})$ to obtain feasibility/optimality cuts $\overline{C}$ and optimal value $\overline{\rho}$

20. update $C \leftarrow C \cup \overline{C}$, refine $\mathcal{D}$ w.r.t $\overline{C}$, and update its longest $r$-path solution $(\vec{x}, z)$ and length $w$

21. until $z = \overline{\rho}$

22. For each node $u \in \text{EXACT}(\mathcal{D})$, add the partial assignment encoding a longest $r$-u path of $\mathcal{D}$ to $\hat{X}$

23. else

24. Go to line 2

25. Return $(\vec{x}^*, z^*)$ and $w^*$
Remark 1. For a selected partial assignment $\hat{x} \in \hat{X}$ in line 3 of Algorithm 1, if the restricted DD $D$ constructed in line 4 models an exact representation of $M_C(\hat{x})$, 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.

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

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 $M$ is formulated as $\max_{x \in \{0,1\}^2} \{ x_1 + x_2 + z \mid x_1 + x_2 \geq 1 \}$, where $z$ represents the objective value of the subproblem $S(\bar{x}_1, \bar{x}_2)$ described by

$$\max_{y \in \mathbb{R}_+^2} \left\{ 2y_1 + y_2 \begin{array}{c} y_1 + y_2 \geq \bar{x}_1 + \bar{x}_2 \\ 0.3y_1 + 0.7y_2 \leq 0.1x_1 + 0.3 \end{array} \right\}.$$  

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

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

The feasibility cuts are of the form $\hat{\pi}_1(x_1 + x_2) - \hat{\pi}_2(0.1x_1 + 0.3) \leq 0$ where $\hat{\pi}$ 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 $D$ 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 $D$, at this iteration, is associated with the solution $(x_1, x_2, z) = (1, 1, M)$ with length $w = 2 + M$. Solving $S(x)$ gives a feasibility cut $0.66x_1 + x_2 \leq 1$. We update the set of cuts $C$ and refine $D$ 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 $(x_1, x_2, z) = (0, 1, M)$ gives the optimal solution $\hat{\pi} = (0, 6.66)$ and the optimal value $\rho = 2$ for the dual subproblem, generating the optimality cut $z \leq 0.66x_1 + 2$ through solving $S(x)$. Refining $D$ 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 $w$ equal to 3.66 and is achieved by $(x_1, x_2, z) = (1, 0, 2.66)$. At this point, we update $w^* = w = 3.66$ and $(x^*, z^*) = (x, z) = (1, 0, 2.66)$. Since $D$ models an exact representation of $\mathcal{M}^C(x)$, it follows from Remark 1 that we can skip creating relaxed DDs, which leads to terminating the algorithm with the optimal solution $(x^*, z^*) = (1, 0, 2.66)$ 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 Section 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 electric 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, generation capacity, demand satisfaction, ramping, 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 generation unit $i \in N = \{1, \ldots, n\}$ is up at time $j \in T = \{1, \ldots, T\}$. 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 exponential 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 an exponential 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 $\ell^i$ (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 $j$.

$$
\min \sum_{i=1}^{n} \sum_{j=1}^{T} \left(c^i_j x^i_j + c^i_j p^i_j + q^i_j \right)
$$

s.t. 

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

$$y^i_j - \bar{y}^i_j = x^i_j - x^i_{j-1} \quad \forall j \in \{1, \ldots, T\}, \quad \forall i \in N \quad (1c)$$

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

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

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

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

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

$$\sum_{i=1}^{n} p^i_j \geq D_j \quad \forall j \in \mathcal{T} \quad (1i)$$

$$\sum_{i=1}^{n} \bar{p}^i_j \geq D_j + R_j \quad \forall j \in \mathcal{T} \quad (1j)$$
\[x_j^i, y_j^i, \bar{y}_j^i \in \{0, 1\} \quad \forall j \in T, \quad \forall i \in N \quad (1k)\]
\[q_j^i, p_j^i, \bar{p}_j^i \geq 0 \quad \forall j \in T, \quad \forall i \in N. \quad (1l)\]

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 \textit{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
\[
\min \left\{ \sum_{i=1}^{n} \sum_{j=1}^{T} (c_{ij}^i x_j^i + q_j^i) \left| (1b) - (1e), (x, y, \bar{y}) \in \{0, 1\}^{3nT}, q \in \mathbb{R}^{nT}, z \in [-\Gamma, \Gamma] \right. \right\}, \quad (2)
\]
where \(\Gamma\) and \(-\Gamma\) are some valid bounds on \(z\) induced from the MIP formulation, and \(z\) represents the objective value of the following subproblem
\[
\min \left\{ \sum_{i=1}^{n} \sum_{j=1}^{T} c_{ij}^i p_j^i \left| (1f) - (1j), (p, \bar{p}) \in \mathbb{R}_{+}^{2nT} \right. \right\}. \quad (3)
\]
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. Section 4.4 presents computational results for this stochastic variant.

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 \((x, y, \bar{y})\) together with continuous variables \((q, z)\), a typical DD would contain all five variable types in its arc layers. It turns out that it suffices to model variables \(x\) 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 \(u \in U_j\) 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 exponential 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 \((\infty, \infty)\) 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 \(D = (U, A, l(.))\) with \(T + 1\) arc layers representing variables \(x_j\) for \(j \in T\), as well as the continuous variable \(z\) at the last layer. Each
arc $a \in A$ 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.

**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 $[D|w(.)]$

1. create the root node with state value $(\infty, \infty)$.

2. forall $j \in T$ and $u \in U_j$ do

3. if $s_u^+ \geq s_u^-$ then

4. create a node $v \in U_{j+1}$ with state value $(s_u^+ + 1, s_u^- + 1)$, and an arc $a \in A_j$ connecting $u$ to $v$ with $l_a = 0$ and $w_a = 0$

5. if $s_u^- \geq \ell$ then

6. create a node $v \in U_{j+1}$ with state value $(1, s_u^- + 1)$, and an arc $a \in A_j$ connecting $u$ to $v$ with $l_a = 1$ and $w_a = c_f + K s_u^-$

7. else

8. create a node $v \in U_{j+1}$ with state value $(s_u^+ + 1, s_u^- + 1)$, and an arc $a \in A_j$ connecting $u$ to $v$ with $l_a = 1$ and $w_a = c_f$

9. if $s_u^+ \geq L$ then

10. create a node $v \in U_{j+1}$ with state value $(s_u^+ + 1, 1)$, and an arc $a \in A_j$ connecting $u$ to $v$ with $l_a = 0$ and $w_a = 0$

11. forall $u \in U_{T+1}$ do

12. create two arcs $a_1, a_2 \in A_{T+1}$ connecting $u$ to the terminal node with $l_{a_1} = w_{a_1} = \Gamma$, $l_{a_2} = w_{a_2} = -\Gamma$.

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 \((\hat{x}, \hat{y}, \hat{y}, \hat{q}, \hat{z})\) of (2) as non-redundant if there is no distinct point \((\hat{x}, \hat{y}, \hat{y}, \hat{q}, \hat{z})\) of (2) with a strictly smaller objective value.

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

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.

**Definition 2.** Consider the following modification of Algorithm 2: (i) each node \(u \in \mathcal{U}_j\) for \(j \in \{1, \ldots, T + 1\}\) is assigned with state values \((s_{u}^+, s_{u}^-, s_{u}^\uparrow)\) with an additional component \(s_{u}^\uparrow\) whose value is initialized and updated similarly to that of \(s_{u}^-\) throughout the iterations of the algorithm; (ii) parameter \(K_{s_{u}^\uparrow}\) is replaced with \(K_{s_{u}^\uparrow}\) 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 \( j \in \{1, \ldots, T + 1\} \), a collection of nodes \( \{u_1, \ldots, u_k\} \) with the property that either \( s^+_{u_i} \geq s^-_{u_i} \) for all \( i \in \{1, \ldots, k\} \), or \( s^+_{u_i} < s^-_{u_i} \) for all \( i \in \{1, \ldots, k\} \) are merged into node \( v \) with state values \( s^+_v = \max_{i \in \{1, \ldots, k\}} s^+_u, s^-_v = \max_{i \in \{1, \ldots, k\}} s^-_u, \) and \( s^=_v = \min_{i \in \{1, \ldots, k\}} s^=_u. \)

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

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 \((x, y, \bar{y}, z)\). As a result, they cannot be directly used to refine the DDs representing the master problem as they only contain variables \((x, z)\); 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 \((y, \bar{y})\). Such a substitution in the subproblem (3) leads to feasibility and optimality cuts that are in the space of \((x, z)\), 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 \( E \) be the feasible region of the UCP described by (1b)–(1l). Define \( G \) to be the feasible region of the UCP where constraints (1f) and (1g) are respectively replaced with

\[
\begin{align*}
p^i_j - p^i_{j-1} &\leq (RU^i - SU^i) x^i_{j-1} + SU^i x^i_j \quad \forall j \in T, \forall i \in N \quad (4a) \\
p^i_{j-1} - p^i_j &\leq (RD^i - SD^i) x^i_j + SD^i x^i_{j-1} \quad \forall j \in T, \forall i \in N. \quad (4b)
\end{align*}
\]

Then, \( E = G \).  \( \square \)
Substituting (1f) and (1g) with (4a) and (4b) in the description of the subproblem (3), we obtain the following dual problem

\[
\begin{align*}
\max & \quad \sum_{j=1}^{T} D_j \psi_j + \sum_{j=1}^{T} (D_j + R_j) \beta_j + \sum_{i=1}^{n} \sum_{j=1}^{T} \bar{x}_{ij}^j \left( m^i \phi_{i,j} - M^i \pi_{i,j} \right) \\
& \quad + \sum_{i=1}^{n} \sum_{j=1}^{T} \left( (SU^i - RU^i) \bar{x}_{ij-1}^j - SU^i \bar{x}_{ij}^j \right) \gamma_{ij} \\
& \quad + \sum_{i=1}^{n} \sum_{j=1}^{T} \left( (SD^i - RD^i) \bar{x}_{ij}^j - SD^i \bar{x}_{ij-1}^j \right) \delta_{ij} \\
\text{s.t.} & \quad \psi_j - \gamma_{ij} + \gamma_{i+1,j} + \delta_{i,j} - \delta_{i,j+1} + \phi_{i,j} - \eta_{i,j} \leq c^j_g \quad \forall j \in \mathcal{T}, \: \forall i \in \mathcal{N} \\
& \quad \beta_j + \eta_{i,j} - \pi_{i,j} \leq 0 \quad \forall j \in \mathcal{T}, \: \forall i \in \mathcal{N} \\
& \quad \psi_j, \beta_j, \gamma_{ij}, \delta_{ij}, \phi_{i,j}, \eta_{i,j}, \pi_{i,j} \geq 0 \quad \forall j \in \mathcal{T}, \: \forall i \in \mathcal{N}.
\end{align*}
\] (5a)

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

\[
0 \geq \sum_{j=1}^{T} D_j \tilde{\psi}_j + \sum_{j=1}^{T} (D_j + R_j) \tilde{\beta}_j + \sum_{i=1}^{n} \sum_{j=1}^{T} \left( m^i \phi_{i,j} - M^i \pi_{i,j} \right) x_{ij}^i \\
+ \sum_{i=1}^{n} \sum_{j=1}^{T} \left( \tilde{\gamma}_{ij} \left( (SU^i - RU^i) x_{ij-1}^j - SU^i x_{ij}^j \right) + \tilde{\delta}_{ij} \left( (SD^i - RD^i) x_{ij}^j - SD^i x_{ij-1}^j \right) \right). 
\] (6)

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

\[
z \geq \sum_{j=1}^{T} D_j \psi_j^* + \sum_{j=1}^{T} (D_j + R_j) \beta_j^* + \sum_{i=1}^{n} \sum_{j=1}^{T} \left( m^i \phi_{i,j}^* - M^i \pi_{i,j}^* \right) x_{ij}^i \\
+ \sum_{i=1}^{n} \sum_{j=1}^{T} \left( \gamma_{ij}^* \left( (SU^i - RU^i) x_{i-1,j}^j - SU^i x_{ij}^j \right) + \delta_{ij}^* \left( (SD^i - RD^i) x_{ij}^j - SD^i x_{ij-1}^j \right) \right). 
\] (7)

### 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 this purpose, we compare the running time of the DD-BD with that of the standard BD algorithm solved by Gurobi.
optimization solver version 9.0.0. We refer to the latter solution method as G-BD. We set the width 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. We capture demand uncertainty by a set of scenarios $\Xi$. At each iteration of both DD-BD and G-BD approaches, we solve all $|\Xi|$ subproblems through Gurobi and add the associated cuts to the master problem. When solving problems with Gurobi, we turn off presolve, heuristics, and cuts for both methods to minimize bias created by solver’s settings. All experiments were conducted on a machine running Windows 10, x64 operating system with Intel® Core i7 processor (1.80 GHz) and 6 GB RAM. The code is written in Microsoft Visual Studio 2015 in C++.

<table>
<thead>
<tr>
<th>Unit</th>
<th>$M$</th>
<th>$m$</th>
<th>$RU$</th>
<th>$RD$</th>
<th>$SU$</th>
<th>$SD$</th>
<th>$e_g$</th>
<th>$c_f$</th>
<th>$C_{SU}^{\text{hot}}$</th>
<th>$C_{SU}^{\text{cold}}$</th>
<th>$t_{\text{cold}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>455</td>
<td>150</td>
<td>225</td>
<td>225</td>
<td>150</td>
<td>150</td>
<td>16.19</td>
<td>1000</td>
<td>4500</td>
<td>9000</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>455</td>
<td>150</td>
<td>225</td>
<td>225</td>
<td>150</td>
<td>150</td>
<td>17.26</td>
<td>970</td>
<td>5000</td>
<td>10000</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>130</td>
<td>20</td>
<td>50</td>
<td>50</td>
<td>20</td>
<td>20</td>
<td>16.60</td>
<td>700</td>
<td>550</td>
<td>1100</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>130</td>
<td>20</td>
<td>50</td>
<td>50</td>
<td>20</td>
<td>20</td>
<td>16.50</td>
<td>680</td>
<td>560</td>
<td>1120</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>162</td>
<td>25</td>
<td>60</td>
<td>60</td>
<td>25</td>
<td>25</td>
<td>19.70</td>
<td>450</td>
<td>900</td>
<td>1800</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>80</td>
<td>20</td>
<td>60</td>
<td>60</td>
<td>20</td>
<td>20</td>
<td>22.26</td>
<td>370</td>
<td>170</td>
<td>340</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>85</td>
<td>25</td>
<td>60</td>
<td>60</td>
<td>25</td>
<td>25</td>
<td>27.74</td>
<td>480</td>
<td>260</td>
<td>520</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>55</td>
<td>10</td>
<td>135</td>
<td>135</td>
<td>10</td>
<td>10</td>
<td>25.92</td>
<td>660</td>
<td>30</td>
<td>60</td>
<td>0</td>
</tr>
</tbody>
</table>

For our experiments, we use benchmark instances to generate data as follows. We consider eight power generation units (Ostrowski et al. 2011, Morales-España et al. 2013); see Table 1. The start-up $SU$ and shut-down $SD$ rates are assumed to be equal to the generators minimum capacity $m$ (Morales-España et al. 2013). At each time period, we consider two values for the start-up cost steps (Ostrowski et al. 2011). If a generator starts up within $t_{\text{cold}}$ hours,
then the start-up cost is $C_{SU}^{hot}$, and it is $C_{SU}^{cold}$ otherwise. We experiment with three classes, $|\Xi| \in \{10, 30, 50\}$, for the number of demand scenarios. Under each scenario class, we consider five categories $(0.5\overline{M}, 0.6\overline{M}, 0.7\overline{M}, 0.8\overline{M}, 0.9\overline{M})$ for lower bound of power demands where $\overline{M}$ denotes the total capacity over all generators, i.e., $\overline{M} = \sum_{i=1}^{n} M^i$. For each scenario class and demand category, we have created 10 instances with random demands (according to the category cut-off) for 24 time periods, representing short-term energy models. Subsequently, we have solved $3 \times 5 \times 10 = 150$ instances for each solution method in total.

Figures 5, 6, and 7 compare the performance of DD-BD and G-BD methods through box and whisker plots under each scenario class and demand category. As evidenced in these figures, the DD-BD method uniformly outperforms the G-BD approach. In particular, the minimum, median, and maximum running times of the DD-BD method is smaller than those of the G-BD approach under all scenario classes and demand categories, providing order of magnitudes time improvement. Another observation is that the variability in running times of the DD-BD method is smaller than that of the G-BD, rendering the DD-BD as a more robust approach. We report the running times of DD-BD and G-BD methods in solving all 150 instances in Tables 2-4 in Appendix C. These preliminary results show a promising potential for the DD-BD method to solve different MIP models in energy applications.

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.
Figure 5  Comparison of the G-BD and DD-BD methods for 10 demand scenarios

Figure 6  Comparison of the G-BD and DD-BD methods for 30 demand scenarios
Figure 7  Comparison of the G-BD and DD-BD methods for 50 demand scenarios
Appendix A: Omitted Proofs

Proof of Theorem 1. Define \( z^1 = \max \{ f(x) \mid x \in \bigcup_{j \in J} \mathcal{X}(P_j^I) \} \) and \( z^2 = \max \{ f(x) \mid x \in \bigcup_{j \in J} \bigcup_{k \in K_j} R_{jk}^I \} \). We first show that \( z^1 = z^2 \). For each \( j \in J \), we write that \( \mathcal{X}(P_j^I) \subseteq \mathcal{X}(\bigcup_{k \in K_j} R_{jk}^I) \subseteq \bigcup_{k \in K_j} R_{jk}^I \), 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 \( \bigcup_{j \in J} \mathcal{X}(P_j^I) \) is a finite set since \( \bigcup_{j \in J} \mathcal{X}(\bigcup_{k \in K_j} R_{jk}^I) \) is finite because of the finiteness of \( J, K_j \) and the set of extreme points of hyper-rectangles \( R_{jk}^I \). Let \( x^* \) be an optimal solution for \( z^2 \). It follows that \( x^* \in \bigcup_{k \in K_j^*, j} R_{jk}^I \), for some \( j^* \in J \). Conditions (i) and (iii) imply that the value of \( x^*_i \) is fixed for coordinates \( i \in N \setminus I \), as it is fixed for \( P_i^* \). Since \( f(x) \) is convex in the unfixed variables \( x_i \), its maximum over \( \bigcup_{k \in K_j^*, j} R_{jk}^I \) occurs at an extreme point \( \tilde{x} \) of \( \text{conv}(\bigcup_{k \in K_j^*, j} R_{jk}^I) \). Condition (iii) implies that \( \tilde{x} \in \mathcal{X}(P_i^*) \), proving that \( z^2 \leq z^1 \). For the second part of the proof, we show that \( z^3 = \max \{ f(x) \mid x \in \mathcal{Q} \} = z^1 = z^2 \). It follows from condition (ii) that \( z^3 \leq z^2 \leq z^4 \) where \( z^4 = \max \{ f(x) \mid x \in \bigcup_{j \in J} P_j^I \} \). Using an argument similar to that given above, we conclude that the optimal value \( z^4 \) is attained at an extreme point of \( P_i^* \) for some \( j^* \in J \), which is also an extreme point of \( \bigcup_{k \in K_j^*, j} R_{jk}^I \). As a result \( z^4 \leq z^2 = z^1 \), proving the result. \( \square \)

Proof of Proposition 1. Consider any point \( \bar{x}_{N \setminus I} \in \text{proj}_{x_{N \setminus I}}(\mathcal{P}) \), and define the indicator function
\[
\delta(x | \bar{x}_{N \setminus I}) = \begin{cases} 
0, & \text{if } x_{N \setminus I} = \bar{x}_{N \setminus I} \\
-\infty, & \text{else.}
\end{cases}
\]
We can use \( \delta(x | \bar{x}_{N \setminus I}) \) as the objective function in the relation \( \max \{ f(x) \mid x \in \mathcal{P} \} = \max \{ f(x) \mid x \in \mathcal{Q} \} \) as it is convex in \( x_i \). It follows that \( \bar{x}_{N \setminus I} \in \text{proj}_{x_{N \setminus I}}(\mathcal{Q}) \). 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. For any point \( \bar{x}_{N \setminus I} \in \text{proj}_{x_{N \setminus I}}(\mathcal{P}) \), define \( \mathcal{P}(\bar{x}_{N \setminus I}) = \mathcal{P} \cap \{ x | x_{N \setminus I} = \bar{x}_{N \setminus I} \} \), and \( \mathcal{Q}(\bar{x}_{N \setminus I}) = \mathcal{Q} \cap \{ x | x_{N \setminus I} = \bar{x}_{N \setminus I} \} \). For any convex function \( f_i(x_i) \) defined in the space of variables \( x_i \), and any point \( \bar{x}_{N \setminus I} \in \text{proj}_{x_{N \setminus I}}(\mathcal{P}) \), we have that
\[
\max \{ f_i(x_i) | x \in \mathcal{P}(\bar{x}_{N \setminus I}) \} = \max \{ \delta(x | \bar{x}_{N \setminus I}) + f_i(x_i) | x \in \mathcal{P} \} \quad (8a)
\]
\[
= \max \{ \delta(x | \bar{x}_{N \setminus I}) + f_i(x_i) | x \in \mathcal{Q} \} \quad (8b)
\]
\[
= \max \{ f_i(x_i) | x \in \mathcal{Q}(\bar{x}_{N \setminus I}) \} \quad (8c)
\]
where (8a) and (8d) follow from the definition of the indicator function above, and (8b) follows from the assumption. Since \( f_i(x_i) \) is convex, the optimal value of (8a) is attained at an extreme point \( \hat{x} \) of \( \mathcal{P}(\bar{x}_{N \setminus I}) \). For each such extreme point, there are infinitely many convex functions \( \hat{f}_i(x_i) \) with \( \hat{x} \) as their unique maximizer over \( \mathcal{P}(\bar{x}_{N \setminus I}) \). For every one of these functions, according to the
chain equalities (8a)–(8d), $Q$ has a point that matches the optimal value. Since $Q$ is finite, it must be that $\bar{x} \in Q(\bar{x}_{N\setminus I})$. This shows that the set of extreme points of $P(\bar{x}_{N\setminus I})$ is finite. Similarly, it can be shown that for any extreme point $\hat{x}$ of $\text{conv}(Q(\bar{x}_{N\setminus I}))$, we have that $\hat{x} \in P(\bar{x}_{N\setminus I})$. As a result, $\text{conv}(P(\bar{x}_{N\setminus I})) = \text{conv}(Q(\bar{x}_{N\setminus I}))$ for every $\bar{x}_{N\setminus I} \in \text{proj}_{x_{N\setminus I}}(P)$. Now, construct sets $P^j_I$ in Theorem 1 as $\text{conv}(P(\bar{x}_{N\setminus I}))$ for every $\bar{x}_{N\setminus I} \in \text{proj}_{x_{N\setminus I}}(P)$, which yields a finite collection. Condition (i) is satisfied as each set $P^j_I$ is restricted at $\{x | x_{N\setminus I} = \bar{x}_{N\setminus I}\}$. Condition (ii) holds since for any extreme point of $P^j_I$, there is a point of $P$ by construction, and since any point $\bar{x} \in P$ satisfies $\bar{x} \in P(\bar{x}_{N\setminus I}) \subseteq \text{conv}(P(\bar{x}_{N\setminus I}))$. For condition (iii), rectangles $R^j_I$ can be considered as the set of extreme points of $P^j_I$, which has been shown to be finite.

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

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

$$S_I(u) = \left\{ x \in \mathbb{R}^n \mid \min_{i \in I} x_i \leq x_i \leq \max_{i \in I} x_i, \quad \forall i \in I \right\}.$$

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

**Proof of Corollary 1.** For the direct implication, assume that $\mathcal{P}$ admits a rectangular decomposition w.r.t. $I$ through sets $P^j_I$ for $j \in J$. Then, the DD that encodes the finite collection of points $\bigcup_{j \in J} \mathcal{X}(P^j_I)$ provides the desired DD representation because of Theorem 1. For the reverse implication, assume that $\mathcal{P}$ is DD-representable w.r.t. $I$ through $\mathcal{D}$. 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 $Q$ is bounded, there are finitely many points $\bar{x} \in \text{proj}_z(P)$. For each such point, it follows from the definition of projection and the boundedness of $Q$ that there exists an interval $[l_{\bar{x}}, u_{\bar{x}}]$ such that $(\bar{x}; y) \in P$ for every $y \in [l_{\bar{x}}, u_{\bar{x}}]$. As a result, we can write that $\mathcal{P} = \bigcup_{x \in \text{proj}_z(P)} R^x_y$ where $R^x_y = \{ (x; y) \in \mathbb{R}^{n+1} | x = x, \ y \in [l_{\bar{x}}, u_{\bar{x}}] \}$. Hyper-rectangles $R^x_y$ 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.

**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 \( \mathcal{X} \). These partial assignments are generated based on longest \( r-u \) paths associated with nodes \( u \) in the exact cut set of relaxed DDs \( \mathcal{D} \), which contain a finite number of paths by definition; see line 22 of the algorithm. Each resulting partial assignment \( \mathcal{a} \) 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 \( \mathcal{X} \) is all possible partial assignments of the solution set of \( \mathcal{M} \) for \( x \) variables. Because the discrete set \( \mathcal{P} \) in the description of \( \mathcal{M} \) is bounded by assumption, we conclude that \( \mathcal{X} \) 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 \( \mathcal{D} \) 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 \( \mathcal{x} \). 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. 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 \((x^*, z^*)\) and \(w^*\) of Algorithm 1 give an optimal solution and optimal value of \( \mathcal{H} \), respectively. To this end, we first prove that \((x^*, z^*)\) 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 \( \mathcal{D} \), after being refined with respect to optimality and feasibility cuts generated from subproblems \( \mathcal{S}(.) \). We write that \((x^*, z^*) \in \text{Sol}(\mathcal{D}) \subseteq \mathcal{M}(\mathcal{C} \mathcal{a}) \subseteq \mathcal{M}\), where the second inclusion holds because \( \mathcal{D} \) is a restricted DD associated with \( \mathcal{M}(\mathcal{C} \mathcal{a}) \) for some \( C \) and \( \mathcal{a} \), and the third inclusion follows from the fact that a feasible solution to \( \mathcal{M}(\mathcal{C} \mathcal{a}) \) is feasible to \( \mathcal{M} \) by definition. Further, the BD implementation 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, \((x^*, z^*)\) satisfies constraints in both master and 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 \((x^*, z^*)\) is indeed an optimal solution to \( \mathcal{H} \). Assume by contradiction that there exists an optimal solution \((\tilde{x}, \tilde{z})\) of \( \mathcal{H} \) with optimal value \( \tilde{w} \) such that \( \tilde{w} > w^* \). There are three cases. (i) Assume that \( \tilde{x} \) is added as a partial assignment to set \( \mathcal{X} \) at some iteration of the algorithm. For the while loop where \( \tilde{x} \) is selected at line 3, all \( x \) variables are fixed for the restricted DD \( \mathcal{D} \). Therefore, \( \tilde{x} \) is used as the input for the subproblem, i.e., we solve \( \mathcal{S}(\tilde{x}) \), which yields the optimal value \( \tilde{z} \) since \((\tilde{x}, \tilde{z})\) is an optimal solution of \( \mathcal{H} \). Since the weights on the arcs of the restricted DD
\( \mathcal{D} \) are set as the coefficients of variables in the linear objective function of \( \mathcal{H} \), the length of the \( r-t \) path of \( \mathcal{D} \) encoded by \((\tilde{x}, \tilde{z})\) 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 \((\tilde{x}, \tilde{z})\) and \( \tilde{w} \). As a result, \((x^*, z^*)\) and \( w^* \) cannot be returned by the algorithm as the output, a contradiction. (ii) Assume that \( \tilde{x} \) is not added as a partial assignment to set \( \hat{X} \) because the algorithm terminates before such a partial assignment is reached in line 22. This implies that there must be a partial assignment \( \hat{x} \in \hat{X} \) with \( \hat{x}_i = x_i \) for \( i = 1, \ldots, j-1 \) for some \( j \in N \) such that the relaxed DD \( \mathcal{D} \) associated with \( \mathcal{M}^C(\hat{x}) \) for some \( C \) is pruned without reaching line 22. The only possibility for this event is that the length \( \tilde{w} \) of the longest \( r-t \) path of \( \mathcal{D} \) must satisfy \( \tilde{w} \leq w^* \), violating the condition in line 17. However, because of the facts that \( \mathcal{D} \) is a relaxed DD associated with \( \mathcal{M}^C(\hat{x}) \), and that \((\tilde{x}, \tilde{z})\) must be a feasible solution to \( \mathcal{M}^C(\hat{x}) \), we conclude that \( \tilde{w} \geq \tilde{w} \). 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 \( \tilde{x} \) is not added as a partial assignment to set \( \hat{X} \) because the longest \( r-u \) path chosen in line 22 deviates from \( \tilde{x} \) for some node \( u \) of the the path associated with \( \tilde{x} \). This implies that there must be a partial assignment \( \hat{x} \in \hat{X} \) with \( \hat{x}_i = x_i \) for \( i = 1, \ldots, j-1 \) for some \( j \in N \) such that the relaxed DD \( \mathcal{D} \) associated with \( \mathcal{M}^C(\hat{x}) \) for some \( C \) has an exact cut set that contains node \( u \) in some layer \( k \geq j \). Let \( \tilde{x} \) be the solution encoding the longest \( r-u \) path of \( \mathcal{D} \) that is chosen in line 22 in place of \( \tilde{x} \). It follows from definition of exact cut set that an extension of \( \tilde{x} \) with components \( \hat{x}_i = \hat{x}_i \) for \( i = k, \ldots, n \) and \( \hat{z} = \hat{z} \) is a feasible solution to \( \mathcal{H} \). However, the above assumption implies that the length \( \tilde{w} \) of the path encoded by \((\tilde{x}, \tilde{z})\) is no less than \( \tilde{w} \). If \( \tilde{w} > \tilde{w} \), then we have reached a contradiction to the assumption that \( \tilde{w} \) is the optimal value of \( \mathcal{H} \). If \( \tilde{w} = \tilde{w} \), we may repeat the contradiction arguments for the new point \((\tilde{x}, \tilde{z})\) 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 \( u \in U \) 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 \((\hat{x}, \hat{z})\) encodes an \( r-t \) path of length \( \hat{c} \) in the equivalence class formed by \( \mathcal{D} \). We construct an extended solution \((\hat{x}, \hat{y}, \hat{y}, \hat{q}, \hat{z})\) of (2) with objective value \( \hat{c} \) as follows. First, we show that \( \hat{x}_j \in \{0,1\} \) for all \( j \in T \) and \( \hat{z} \in [-\Gamma, \Gamma] \), thereby satisfying domain constraints in (2). The definition of equivalence class implies that, for each \( j \in T \), there exists a node
pair \((u, v) \in \mathcal{A}_j \times \mathcal{A}_{j+1}\) such that \(\hat{x}_j \in \left[l_{(u,v)}^{\min}, l_{(u,v)}^{\max}\right]\), 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 \(l_{(u, v)}^{\min} = l_{(u, v)}^{\max} \in \{0, 1\}\). The argument for \(\hat{z} \in [-\Gamma, \Gamma]\) 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 \(\hat{y}_j = 1\) if \(\hat{x}_{j-1} = 0\) and \(\hat{x}_j = 1\), and \(\hat{y}_j = 0\) otherwise. Similarly, define \(\hat{\bar{y}}_j = 1\) if \(\hat{x}_{j-1} = 1\) and \(\hat{x}_j = 0\), and \(\hat{\bar{y}}_j = 0\) 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 \((\hat{x}, \hat{z})\), and define \(\hat{q}_j = K_{s_u^+}\) if \(\hat{x}_j = 1\) and \(\hat{x}_{j-1} = 0\), and \(\hat{q}_j = 0\) 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 exponential, we have that \(\max_{k=1, \ldots, s_u^-} K_k = K_{s_u^-}\). Further, since \(\hat{q}_j\) 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 \((\hat{x}, \hat{y}, \hat{\bar{y}}, \hat{q}, \hat{z})\). 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., \(\hat{x}_{j^*} = 0\) and there exists \(\bar{j} \in \{j^* - L + 1, \ldots, j^*\}\) such that \(\hat{y}_{\bar{j}} = 1\). It follows from construction that \(\hat{x}_{\bar{j}} = 1\) and \(\hat{x}_{\bar{j}-1} = 0\). Let \((u, v) \in \mathcal{U}_j \times \mathcal{U}_{j+1}\) be the nodes at layers \(\bar{j}\) and \(\bar{j} + 1\) of the \(r\)-\(t\) path encoding \((\hat{x}, \hat{z})\). 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 \((\hat{x}_{\bar{j}} = 1)\), we have \(s_u^+ = 1\); 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 \(\bar{j} + L, \ldots, T\). Since \(\bar{j} \geq j^* - L + 1\) by definition, we obtain that \(j^* < \bar{j} + L\), and hence \(\hat{x}_{j^*}\) 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 \(j^* - L + 1, \ldots, j^*\). It is easy to verify that there exists a layer \(\tilde{j} \in \{j^* - L + 1, \ldots, j^*\}\) 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 \((\hat{x}, \hat{y}, \hat{\bar{y}}, \hat{q}, \hat{z})\) is feasible to (2). We next show that the length of the \(r\)-\(t\) path encoding \((\hat{x}, \hat{z})\) matches the objective value of \((\hat{x}, \hat{y}, \hat{\bar{y}}, \hat{q}, \hat{z})\). The proof follows from considering the contribution of three terms in the objective function of (2). First, the contribution of each variable assignment \(\hat{x}_j = 1\) 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 $(\dot{x}, \dot{z})$, 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 $D$.

For the reverse implication, assume that $(\dot{x}, \dot{y}, \dot{y}, \dot{q}, \dot{z})$ is a non-redundant solution of (2) with objective value $\dot{c}$, we show that the projected point $(\dot{x}, \dot{z})$ encodes an $r$-$t$ path of length $\dot{c}$ in the equivalence class formed by $D$. Assume by contradiction that no such path exists in the equivalence class of $D$. Therefore, there exists an $r$-$t$ path $P$ of $D$ with associated point $(\dot{x}, \dot{z})$ and a layer number $j^* \in T \setminus \{T\}$ such that $\dot{x}_j = \dot{x}_j$ for $j \in \{1, \ldots, j^*\}$ and the node $u$ at layer $j^* + 1$ of $P$ 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 $s^+_v \geq s^-_u$ as $\dot{x}_{j^*} = \dot{x}_{j^*} = 0$. 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 $\dot{x}_j = \dot{x}_j = 1$ for $j = j^* - L + 1, \ldots, j^*$. Let $v$ be the node at layer $j^* - L + 1$ of $P$. Since $\dot{x}_j = 1$ for $j = j^* - L + 1, \ldots, j^*$, it follows from line 8 of Algorithm 2 that $s^+_v = s^+_u + L - 1 \geq L$ as $s^+_u \geq 1$. 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 $\dot{x}_{j^*} = 0$ 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 $D$ as $-\Gamma$ and $\Gamma$ are used as label values of the pair of arcs connecting every node of layer $T + 1$ to the terminal node in $D$. We conclude that $(\dot{x}, \dot{z})$ encodes an $r$-$t$ path of length $\dot{c}$ in the equivalence class formed by $D$. To show that the objective value $\dot{c}$ is equal to the length of the $r$-$t$ path encoding $(\dot{x}, \dot{z})$, we note that $\dot{q}_j = K^1_{s_h}$, where $h$ is the node at layer $j$ of the $r$-$t$ path, since otherwise $(\dot{x}, \dot{y}, \dot{y}, \dot{q}, \dot{z})$ 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.

**Proof of Proposition 3.** We need to show that for each path $\tilde{P}$ of $\tilde{D}$ with encoding point $(\tilde{x}, \tilde{z})$, there exists a path $\tilde{P}$ of $\tilde{D}$ with encoding point $(\tilde{x}, \tilde{z})$ such that $\tilde{x} = \tilde{x}$, $\tilde{z} = \tilde{z}$, and $w(\tilde{P}) \geq w(\tilde{P})$, where $w(.)$ represent the length of the path. Define $\tilde{P}^k$ to be the path composed of the first $k$ arcs of $\tilde{P}$ for $k \in \mathcal{T}$. We prove the result by induction on the size of $k$, i.e., we show that for each $\tilde{P}^k$ with encoding point $\tilde{x}^k$, there exists a path $\tilde{P}^k$ of $\tilde{D}$ with encoding point $\tilde{x}^k$ such that $\tilde{x}^k = \tilde{x}^k$ and $w(\tilde{P}^k) \geq w(\tilde{P}^k)$. 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 $\Gamma$ in both $\tilde{D}$ and $\tilde{D}$,
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 \( \mathcal{D} \) 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 \( \tilde{P}^{k-1} \) of \( \tilde{\mathcal{D}} \) with encoding point \( \tilde{x}^{k-1} \) such that \( \tilde{x}^{k-1} = \tilde{x}^{k-1} \) and \( w(\tilde{P}^{k-1}) \geq w(\tilde{P}^{k-1}) \). We show that there exists a path \( \tilde{P}^k \) of \( \tilde{\mathcal{D}} \) with encoding point \( \tilde{x}^k \) such that \( \tilde{x}^k = \tilde{x}^k \) and \( w(\tilde{P}^k) \geq w(\tilde{P}^k) \). There are four cases. (i) Assume that \( \tilde{x}^{k-1}_k = 0 \) and \( \tilde{x}^{k}_k = 0 \). Note that \( \tilde{x}^{k-1}_k = \tilde{x}^{k-1}_k = 0 \) as they are defined over the same path \( \tilde{P} \). It follows from the induction hypothesis that \( \tilde{x}^{k-1}_k = \tilde{x}^{k-1}_k = 0 \). Let \( u \) be the node at node layer \( k \) of the path \( \tilde{P}^{k-1} \). 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 \( \tilde{P}^{k-1} \) yields the desired path \( \tilde{P}^k \), as the contribution to the objective function is 0 for this variable assignment in both paths \( \tilde{P}^k \) and \( \tilde{P}^k \). For the second case, assume that \( u \) is a merged node. Let \( \{v_1, \ldots, v_r\} \) 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 \( s^+_{v_i} \geq s^-_{v_i} \). It follows from the merging equations that \( s^+_i = \max_i, s^+_i \geq \max_i, s^-_{v_i} = s^-_u \). Therefore, an argument similar to that of the first case yields the desired result. (ii) Assume that \( \tilde{x}^{k-1}_k = 0 \) and \( \tilde{x}^{k}_k = 1 \). Let \( h \) be the node at node layer \( k \) of \( \tilde{P}^k \). 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 \( \tilde{x}^{k-1}_{s^-_h} = 1 \) and \( \tilde{x}^{k-1}_{s^-_h} = 0 \) for all \( i = 1, \ldots, s^-_h \). It follows from the induction hypothesis that \( \tilde{x}^{k-1}_{s^-_h} = 1 \) and \( \tilde{x}^{k-1}_{s^-_h} = 0 \) for all \( i = 1, \ldots, s^-_h \). Let \( u \) and \( v \) be the nodes at node layers \( k - s^-_h + 1 \) and \( k \) of \( \tilde{P}^{k-1} \), 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 \( \{u_1, \ldots, u_r\} \) 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 \( \tilde{P}^{k-1} \), we conclude that \( s^-_u \geq s^-_u + s^-_h - 1 \geq s^-_h \) and \( s^-_u \leq s^-_u + s^-_h - 1 \leq s^-_h \), 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 $\tilde{P}^{k-1}$ yields the desired path $\tilde{P}^k$. (iii) Assume that $\tilde{x}_{k-1}^k = 1$ and $\tilde{x}_k^k = 1$. The result is obtained by using an argument similar to that of case (i) due to symmetry. (iv) Assume that $\tilde{x}_{k-1}^k = 1$ and $\tilde{x}_k^k = 0$. 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 $j \in T$. (i) Assume that $x_{j-1}^i = x_j^i = 0$. It follows from constraints (1c) and (1d) that $y_j^i = \tilde{y}_j^i = 0$. 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 $x_{j-1}^i = 0$ and $x_j^i = 1$. It follows from constraint (1c) that $y_j^i = 1$ and $\tilde{y}_j^i = 0$. In this case, the RHS of both constraints (1f) and (4a) reduce to $SU_i$. Further, inequality (1h) implies that $p_{j-1}^i = 0$ as $x_{j-1}^i = 0$. As a result, both constraints (1g) and (4b) become redundant in the description of $E$ and $G$, since their LHS is a non-positive quantity $-p_j^i \leq 0$ and their RHS are non-negative values as $RD^i \geq 0$ and $RD^i - SD^i \geq 0$ by assumption. (iii) Assume that $x_{j-1}^i = 1$ and $x_j^i = 0$. It follows from constraint (1c) that $y_j^i = 0$ and $\tilde{y}_j^i = 1$. In this case, the RHS of both constraints (1g) and (4b) reduce to $SD^i$. Further, inequality (1h) implies that $p_j^i = 0$ as $x_j^i = 0$. As a result, both constraints (1f) and (4a) become redundant in the description of $E$ and $G$, since their LHS is a non-positive quantity $-p_{j-1}^i \leq 0$ and their RHS are non-negative values as $RU^i \geq 0$ and $RU^i - SU^i \geq 0$ by assumption. (iv) Assume that $x_{j-1}^i = x_j^i = 1$. It follows from constraints (1c) and (1e) that $y_j^i = \tilde{y}_j^i = 0$. 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 \( \max \{z \mid x \in \mathcal{P}\} \) with a bounded \( \mathcal{P} \subseteq \mathbb{Z}^n \). Consider \( D = (\mathcal{U}, \mathcal{A}, l(\cdot)) \) 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 \( z \leq \alpha^j x + \alpha^j_0 \), 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 \( z \leq \alpha^j x + \alpha^j_0 \) 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 \( \max \{z \mid x \in \mathcal{P}, \ z \leq \alpha^j x + \alpha^j_0, \forall j \in J\} \).

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

**Data:** a DD \( D = (\mathcal{U}, \mathcal{A}, l(\cdot)) \) and a set of optimality cuts \( z \leq \alpha^j x + \alpha^j_0 \), for \( j \in J \)

1. initialize the root node reward as \( r^j(s) = \alpha^j_0 \) 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 \( r^j(v) = r^j(u) + \alpha^j_i l_a \) 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 \( r(u) = \min_{j \in J} \{r^j(u) + \alpha^j_n l_a\} \) where \( a \in \mathcal{A}_n \) is the unique arc connecting \( u \) to the terminal \( t \)

6. compute the final reward \( r^* = \max_{u \in \mathcal{U}_n} r_u \)

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.

Example 4. Consider the final iteration of a BD algorithm with a master problem \( \max \{ z | x \in P \} \)
where \( P \subseteq \mathbb{Z}^n \) is represented by DD \( D_1 \) of Figure 8a. Assume that the following two optimality
cuts are added

\[
\begin{align*}
  z &\leq 3x_1 + 2x_2 \\
  z &\leq -3x_1 - 5x_2 + 3
\end{align*}
\]

(9a)

(9b)

It is clear that both points \((0,0)\) and \((1,0)\) give an optimal solution with optimal value \( z^* = 0 \). Since
\( D_1 \) 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 \( \mathcal{M} = \max \{ z | x \in P, z \in [-M, M] \} \) where valid bounds are imposed on \( z \) variable. A restricted DD \( D_2 \) 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.

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 $\max \{ z \mid x \in \mathcal{P} \}$ 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 $z \leq \alpha^j x + \alpha_0^j$, for $j \in J$, represent the set of optimality cuts to be added at the current iteration of the BD algorithm. Define $\tilde{\mathcal{D}} = (\tilde{\mathcal{U}}, \tilde{\mathcal{A}}, \tilde{l}(\cdot))$ 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 $\tilde{\mathcal{D}}$. 

Figure 8  DD representation of the master problem of Example 4

(a) $\mathcal{D}_1$ without continuous variable
(b) $\mathcal{D}_2$ with continuous variable

Figure 9  Separation of optimality cuts for the master problem of Example 4

(a) Separation wrt (9a)
(b) Separation wrt (9b)
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 $P = (a_1, \ldots, a_n) \in \bar{\mathcal{A}}_1 \times \ldots \times \bar{\mathcal{A}}_n$ from the root node to $v$, it follows from the assumption that the threshold diagrams have the same structure as $\mathcal{D}$, with the difference that the upper bound label on arcs connecting $v$ to the terminal is computed as $M^j(v) = \sum_{i=1}^n \alpha^j l_{a_i} + \alpha^j_0$ 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. □
Appendix C: Omitted Tables for Numerical Experiments

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Running times (in seconds) of the G-BD and DD-BD methods in solving instances of the stochastic UCP with 10 demand scenarios under 5 demand categories</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Instance</td>
</tr>
<tr>
<td>$D_j \geq 0.5M$</td>
<td>G-BD</td>
</tr>
<tr>
<td></td>
<td>DD-BD</td>
</tr>
<tr>
<td></td>
<td>Instance</td>
</tr>
<tr>
<td>$D_j \geq 0.6M$</td>
<td>G-BD</td>
</tr>
<tr>
<td></td>
<td>DD-BD</td>
</tr>
<tr>
<td></td>
<td>Instance</td>
</tr>
<tr>
<td>$D_j \geq 0.7M$</td>
<td>G-BD</td>
</tr>
<tr>
<td></td>
<td>DD-BD</td>
</tr>
<tr>
<td></td>
<td>Instance</td>
</tr>
<tr>
<td>$D_j \geq 0.8M$</td>
<td>G-BD</td>
</tr>
<tr>
<td></td>
<td>Instance</td>
</tr>
<tr>
<td>$D_j \geq 0.9M$</td>
<td>G-BD</td>
</tr>
<tr>
<td></td>
<td>DD-BD</td>
</tr>
</tbody>
</table>
Table 3  Running times (in seconds) of the G-BD and DD-BD methods in solving instances of the stochastic UCP with 30 demand scenarios under 5 demand categories

<table>
<thead>
<tr>
<th>$D_j \geq 0.5M$</th>
<th>Instance</th>
<th>0.5_1</th>
<th>0.5_2</th>
<th>0.5_3</th>
<th>0.5_4</th>
<th>0.5_5</th>
<th>0.5_6</th>
<th>0.5_7</th>
<th>0.5_8</th>
<th>0.5_9</th>
<th>0.5_10</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-BD</td>
<td>163.48</td>
<td>152.86</td>
<td>103.87</td>
<td>77.14</td>
<td>116.57</td>
<td>95.01</td>
<td>168.37</td>
<td>79.21</td>
<td>95.52</td>
<td>131.65</td>
<td></td>
</tr>
<tr>
<td>DD-BD</td>
<td>59.54</td>
<td>82.63</td>
<td>70.22</td>
<td>63.42</td>
<td>64.52</td>
<td>88.84</td>
<td>42.20</td>
<td>69.17</td>
<td>67.41</td>
<td>51.73</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D_j \geq 0.6M$</th>
<th>Instance</th>
<th>0.6_1</th>
<th>0.6_2</th>
<th>0.6_3</th>
<th>0.6_4</th>
<th>0.6_5</th>
<th>0.6_6</th>
<th>0.6_7</th>
<th>0.6_8</th>
<th>0.6_9</th>
<th>0.6_10</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-BD</td>
<td>97.45</td>
<td>82.34</td>
<td>94.54</td>
<td>82.32</td>
<td>113.03</td>
<td>104.51</td>
<td>150.08</td>
<td>140.01</td>
<td>103.89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DD-BD</td>
<td>57.11</td>
<td>64.87</td>
<td>50.37</td>
<td>41.09</td>
<td>36.75</td>
<td>49.57</td>
<td>79.59</td>
<td>54.52</td>
<td>37.66</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D_j \geq 0.7M$</th>
<th>Instance</th>
<th>0.7_1</th>
<th>0.7_2</th>
<th>0.7_3</th>
<th>0.7_4</th>
<th>0.7_5</th>
<th>0.7_6</th>
<th>0.7_7</th>
<th>0.7_8</th>
<th>0.7_9</th>
<th>0.7_10</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-BD</td>
<td>66.18</td>
<td>106.1</td>
<td>113.63</td>
<td>88.47</td>
<td>111.38</td>
<td>101.78</td>
<td>100.52</td>
<td>99.18</td>
<td>98.75</td>
<td>86.91</td>
<td></td>
</tr>
<tr>
<td>DD-BD</td>
<td>58.72</td>
<td>56.84</td>
<td>57.07</td>
<td>79.47</td>
<td>63.86</td>
<td>34.39</td>
<td>73.20</td>
<td>32.35</td>
<td>60.21</td>
<td>50.98</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D_j \geq 0.8M$</th>
<th>Instance</th>
<th>0.8_1</th>
<th>0.8_2</th>
<th>0.8_3</th>
<th>0.8_4</th>
<th>0.8_5</th>
<th>0.8_6</th>
<th>0.8_7</th>
<th>0.8_8</th>
<th>0.8_9</th>
<th>0.8_10</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-BD</td>
<td>113.33</td>
<td>80.55</td>
<td>82.09</td>
<td>51.76</td>
<td>65.09</td>
<td>78.35</td>
<td>92.72</td>
<td>94.44</td>
<td>98.54</td>
<td>125.39</td>
<td></td>
</tr>
<tr>
<td>DD-BD</td>
<td>32.29</td>
<td>70.01</td>
<td>78.56</td>
<td>48.88</td>
<td>43.92</td>
<td>31.83</td>
<td>44.65</td>
<td>39.66</td>
<td>37.47</td>
<td>62.29</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D_j \geq 0.9M$</th>
<th>Instance</th>
<th>0.9_1</th>
<th>0.9_2</th>
<th>0.9_3</th>
<th>0.9_4</th>
<th>0.9_5</th>
<th>0.9_6</th>
<th>0.9_7</th>
<th>0.9_8</th>
<th>0.9_9</th>
<th>0.9_10</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-BD</td>
<td>71.55</td>
<td>91.08</td>
<td>57.48</td>
<td>74.46</td>
<td>89.11</td>
<td>60.62</td>
<td>75.45</td>
<td>106.81</td>
<td>94.26</td>
<td>63.85</td>
<td></td>
</tr>
<tr>
<td>DD-BD</td>
<td>41.36</td>
<td>54.06</td>
<td>54.86</td>
<td>30.30</td>
<td>52.36</td>
<td>48.78</td>
<td>56.27</td>
<td>54.98</td>
<td>61.36</td>
<td>30.21</td>
<td></td>
</tr>
</tbody>
</table>
### Table 4

Running times (in seconds) of the G-BD and DD-BD methods in solving instances of the stochastic UCP with 50 demand scenarios under 5 demand categories

<table>
<thead>
<tr>
<th>Instance</th>
<th>$D_j \geq 0.5M$</th>
<th>$D_j \geq 0.6M$</th>
<th>$D_j \geq 0.7M$</th>
<th>$D_j \geq 0.8M$</th>
<th>$D_j \geq 0.9M$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>G-BD</td>
<td>G-BD</td>
<td>G-BD</td>
<td>G-BD</td>
<td>G-BD</td>
</tr>
<tr>
<td></td>
<td>90.06</td>
<td>95.61</td>
<td>158.42</td>
<td>137.68</td>
<td>82.53</td>
</tr>
<tr>
<td></td>
<td>139.76</td>
<td>148.35</td>
<td>89.77</td>
<td>89.45</td>
<td>91.03</td>
</tr>
<tr>
<td></td>
<td>162.30</td>
<td>76.69</td>
<td>122.25</td>
<td>136.22</td>
<td>95.05</td>
</tr>
<tr>
<td></td>
<td>159.26</td>
<td>103.38</td>
<td>164.50</td>
<td>136.22</td>
<td>104.87</td>
</tr>
<tr>
<td></td>
<td>124.53</td>
<td>163.90</td>
<td>163.43</td>
<td>99.94</td>
<td>112.01</td>
</tr>
<tr>
<td></td>
<td>101.50</td>
<td>133.34</td>
<td>170.96</td>
<td>87.80</td>
<td>74.50</td>
</tr>
<tr>
<td></td>
<td>143.68</td>
<td>169.19</td>
<td>175.04</td>
<td>117.60</td>
<td>92.15</td>
</tr>
<tr>
<td></td>
<td>146.56</td>
<td>113.07</td>
<td>99.20</td>
<td>119.55</td>
<td>91.15</td>
</tr>
<tr>
<td></td>
<td>172.08</td>
<td>128.93</td>
<td>158.58</td>
<td>133.11</td>
<td>96.55</td>
</tr>
<tr>
<td></td>
<td>123.40</td>
<td>118.40</td>
<td>155.32</td>
<td>121.98</td>
<td>85.88</td>
</tr>
<tr>
<td></td>
<td>80.94</td>
<td>59.38</td>
<td>66.04</td>
<td>81.20</td>
<td>45.69</td>
</tr>
<tr>
<td></td>
<td>74.79</td>
<td>92.78</td>
<td>76.50</td>
<td>52.14</td>
<td>37.78</td>
</tr>
<tr>
<td></td>
<td>108.57</td>
<td>44.93</td>
<td>84.05</td>
<td>72.51</td>
<td>51.01</td>
</tr>
<tr>
<td></td>
<td>125.04</td>
<td>91.79</td>
<td>68.19</td>
<td>59.98</td>
<td>40.45</td>
</tr>
<tr>
<td></td>
<td>103.28</td>
<td>79.72</td>
<td>84.85</td>
<td>67.82</td>
<td>45.99</td>
</tr>
<tr>
<td></td>
<td>70.94</td>
<td>63.68</td>
<td>62.43</td>
<td>78.42</td>
<td>47.80</td>
</tr>
<tr>
<td></td>
<td>73.12</td>
<td>105.64</td>
<td>76.12</td>
<td>71.62</td>
<td>72.06</td>
</tr>
<tr>
<td></td>
<td>76.66</td>
<td>100.29</td>
<td>82.27</td>
<td>94.08</td>
<td>60.94</td>
</tr>
<tr>
<td></td>
<td>64.24</td>
<td>91.44</td>
<td>92.30</td>
<td>87.86</td>
<td>50.05</td>
</tr>
<tr>
<td></td>
<td>109.22</td>
<td>56.93</td>
<td>93.73</td>
<td>76.52</td>
<td>67.62</td>
</tr>
</tbody>
</table>
Acknowledgments

This project is sponsored in part by the Iowa Energy Center, Iowa Economic Development Authority and its utility partners.

References


