Stochastic Dual Dynamic Integer Programming

Jikai Zou* Shabbir Ahmed* Xu Andy Sun*

March 27, 2017

Abstract

Multistage stochastic integer programming (MSIP) combines the difficulty of uncertainty, dynamics, and non-convexity, and constitutes a class of extremely challenging problems. A common formulation for these problems is a dynamic programming formulation involving nested cost-to-go functions. In the linear setting, the cost-to-go functions are convex polyhedral, and decomposition algorithms, such as nested Benders' decomposition and its stochastic variant - Stochastic Dual Dynamic Programming (SDDP) - that proceed by iteratively approximating these functions by cuts or linear inequalities, have been established as effective approaches. It is difficult to directly adapt these algorithms to MSIP due to the nonconvexity of integer programming value functions. In this paper we propose an extension to SDDP – called stochastic dual dynamic integer programming (SDDiP) – for solving MSIP problems with binary state variables. The crucial component of the algorithm is a new class of cuts, termed Lagrangian cuts, derived from a Lagrangian relaxation of a specific reformulation of the subproblems in each stage, where local copies of state variables are introduced. We show that the Lagrangian cuts satisfy a tightness condition and provide a rigorous proof of the finite convergence of SDDiP with probability one. We show that, under fairly reasonable assumptions, an MSIP problem with general state variables can be approximated by one with binary state variables to desired precision with only a modest increase in problem size. Thus our proposed SDDiP approach is applicable to very general classes of MSIP problems. Extensive computational experiments on three classes of real-world problems, namely electric generation expansion, financial portfolio management, and network revenue management, show that the proposed methodology is very effective in solving large-scale, multistage stochastic integer optimization problems.

Keywords: multistage stochastic integer programming, binary state variables, nested decomposition, stochastic dual dynamic programming

*H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA. E-mail: jikai.zou@gatech.edu, shabbir.ahmed@isye.gatech.edu, andy.sun@isye.gatech.edu

1 Introduction

This paper develops effective decomposition algorithms for a large class of multistage stochastic integer programming problems. In this section, we first discuss the problem class of interest, discuss related prior work, and provide a summary of our contributions.

1.1 Multistage Stochastic Programming

Multistage stochastic programming is a framework for sequential decision making under uncertainty where the decision space is typically high dimensional and involves complicated constraints, and the uncertainty is modeled by general stochastic processes. To describe a generic formulation for a multistage stochastic program, let us start with a canonical deterministic optimization problem with T stages:

$$\min_{(x_1,y_1),\ldots,(x_T,y_T)} \left\{ \sum_{t=1}^{T} f_t(x_t, y_t) : \ (x_{t-1}, x_t, y_t) \in X_t, \ \forall\, t = 1, \ldots, T \right\}.$$

In the above formulation we explicitly distinguish two sets of decision variables in each stage, namely, the *state* variable, denoted by x_t, which links successive stages, and the *local* or *stage* variable, denoted by y_t, which is only contained in the subproblem at stage t. This form is without loss of generality since any multistage optimization problem can be formulated in this form by introducing additional constraints and variables. Note that, for notational convenience, the above formulation includes variable x_0 which is assumed to be fixed. The function f_t and the set X_t denote the objective and constraints associated with stage t. We focus on the mixed-integer linear setting where the objective function f_t is linear, and the constraint system X_t is of the form

$$B_t x_{t-1} + A_t x_t + C_t y_t \geq b_t$$

along with integrality restrictions on a subset of the variables. The data required in stage t is $\xi_t := (f_t, X_t)$ where, with some notational abuse, we have used f_t and X_t to denote the data for the objective f_t and constraints in X_t. Let us denote the feasible region of the stage t problem by $F_t(x_{t-1}, \xi_t)$ which depends on the decision in stage $t-1$ and the information ξ_t available in stage t. Suppose now the data $(\boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_T)$ is uncertain and evolves according to a known stochastic process. We use $\boldsymbol{\xi}_t$ to denote the random data vector in stage t and ξ_t to denote a specific realization. Similarly, we use $\boldsymbol{\xi}_{[t,t']}$ denote the sequence of random data vectors corresponding to stages t through t' and $\xi_{[t,t']}$ to denote a specific realization of this sequence of random vectors. The decision dynamics is as follows: in stage t we first observe the data realization ξ_t and then take an action (x_t, y_t) depending on the previous stage decision x_{t-1} (also known as *state*) and the observed data ξ_t to optimize the *expected* future cost. A formulation for this multistage stochastic programming (MSP) problem is:

$$\min_{(x_1,y_1)\in F_1} \left\{ f_1(x_1,y_1) + \mathbb{E}_{\boldsymbol{\xi}_{[2,T]}|\xi_{[1,1]}} \left[\min_{(x_2,y_2)\in F_2(x_1,\xi_2)} \left\{ f_2(x_2,y_2,\xi_2) + \cdots \right.\right.\right.$$
$$\left.\left.\left. + \mathbb{E}_{\boldsymbol{\xi}_{[T,T]}|\xi_{[1,T-1]}} \left[\min_{(x_T,y_T)\in F_T(x_{T-1},\xi_T)} \left\{ f_T(x_T,y_T,\xi_T) \right\} \right] \right\} \right] \right\},$$

where $\mathbb{E}_{\boldsymbol{\xi}_{[t,T]}|\xi_{[1,t-1]}}$ denotes the expectation operation in stage t with respect to the conditional distribution of $\boldsymbol{\xi}_{[t,T]}$ given realization $\xi_{[1,t-1]}$ in stage $t-1$. Depending on whether integer decisions are present, these problems are referred to as multistage stochastic linear programming (MSLP) or multistage stochastic integer programming (MSIP) problems.

Computational approaches for MSP are based on approximating the stochastic process $(\boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_T)$ by a process having finitely many realizations in the form of a scenario tree [see e.g., 71]. Such an approximation may be constructed by Monte Carlo methods as in the sample average approximation (SAA) approach or various other constructive methods [46; 74; 61; 43; 64; 39]. Let $\mathcal{T}$ be the scenario tree associated with the underlying stochastic process. There are T levels corresponding to the T decision-making stages and the set of nodes in stage t is denoted by $\mathcal{S}_t$. The root node in stage 1 is labelled 1, i.e., $\mathcal{S}_1 = \{1\}$. Each node n

in stage $t > 1$ has a unique parent node $a(n)$ in stage $t-1$. We denote the stage containing node n by $t(n)$. The set of children nodes of a node n is denoted by $\mathcal{C}(n)$. The set of nodes on the unique path from node 1 to node n, including node n, is denoted by $\mathcal{P}(n)$. A node $n \in \mathcal{S}_t$ represents a state of the world in stage t and corresponds to the information sequence $\{\xi_m = (f_m, X_m)\}_{m\in\mathcal{P}(n)}$. The total probability associated with node n is denoted as p_n, which is the probability of realization of the $t(n)$-period data sequence $\{\xi_m\}_{m\in\mathcal{P}(n)}$. Each node in the final stage $\mathcal{S}_T$ corresponds to a realization of the data for the full planning horizon, i.e., all T periods, and is called a scenario. For $m \in \mathcal{T} \setminus \{1\}$ and $n = a(m)$, $q_{nm} := p_m/p_n$ is the conditional probability of transitioning from node n to node m. Since the decisions in a stage are taken after observing the data realization we associate the decisions to the nodes of the tree. The resulting formulation, called the *extensive form*, is

$$\min_{x_n, y_n} \left\{ \sum_{n\in\mathcal{T}} p_n f_n(x_n, y_n) : \ (x_{a(n)}, x_n, y_n) \in X_n \ \forall\, n \in \mathcal{T} \right\}. \tag{1.1}$$

While the above formulation is a deterministic optimization problem, it has very large scale as the size of the scenario tree grows exponentially with dimension of the uncertain parameters and the number of stages. An alternative to the extensive form (1.1) is to formulate the MSP problem via the following dynamic programming (DP) equations

$$\min_{x_1, y_1} \left\{ f_1(x_1, y_1) + \sum_{m\in\mathcal{C}(1)} q_{1m} Q_m(x_1) : \ (x_{a(1)}, x_1, y_1) \in X_1 \right\} \tag{1.2}$$

where for each node $n \in \mathcal{T} \setminus \{1\}$

$$Q_n(x_{a(n)}) = \min_{x_n, y_n} \left\{ f_n(x_n, y_n) + \sum_{m\in\mathcal{C}(n)} q_{nm} Q_m(x_n) : \ (x_{a(n)}, x_n, y_n) \in X_n \right\}. \tag{1.3}$$

We will refer to $Q_n(\cdot)$ as the optimal value function (of $x_{a(n)}$) at node n and denote the function $\mathcal{Q}_n(\cdot) := \sum_{m\in\mathcal{C}(n)} q_{nm} Q_m(\cdot)$ as the expected cost-to-go function at node n.

1.2 Prior Work

Multistage stochastic programming has found applications in a variety of sectors. In the energy sector, a classical success story is hydrothermal generation scheduling in Brazil [62; 63] involving the month-to-month planning of power generation of a system of hydro and thermal plants to meet energy demand in the face of stochastic water inflows into the hydro-reservoirs [see also 21; 66; 78]. Numerous other applications in energy have been proposed since. Examples include long term capacity planning of generation and transmission systems [5; 8], day-ahead generation scheduling (unit commitment) [82; 73; 10; 20; 51], planning and operation of renewable energy systems [44; 30; 60; 17], management of electricity storage systems [52; 53], etc. In finance, MSP has been applied to portfolio optimization to maximize the expected return while controlling the risk, as well as in asset-liability management [see e.g., 16; 47; 55; 25; 18; 35]. Beyond energy and finance, multistage stochastic programming has found applications in manufacturing, services, and natural resources [27; 80; 84; 24; 4; 3; 49; 54; 79; 36, etc.]. Motivated by its application potential, there has been a great deal of research on multistage stochastic programming. Major progress has been made on theoretical issues such as structure, complexity, and approximability, as well as on effective decomposition algorithms. Much of the progress, however, has been restricted to the *linear* setting, i.e. MSLP.

In MSLP, the value function $Q_n(\cdot)$ defined in (1.3) and therefore the cost-to-go function $\mathcal{Q}_n(\cdot)$ is piece-wise linear and convex. This allows for these functions to be under approximated by linear cuts as in nested Benders' or L-shaped decomposition [14]. This algorithm approximates the convex cost-to-go functions by adding Benders' cuts, and converges in finite steps to an optimal solution. When the scenario tree is large, however, it may be computationally impractical to solve the problem using nested Benders decomposition. Often the underlying stochastic process and the constructed scenario tree is stage-wise independent, i.e., for any two nodes n and n' in $\mathcal{S}_t$ the set of children nodes $\mathcal{C}(n)$ and $\mathcal{C}(n')$ are defined by identical data and conditional probabilities. Then the value functions and expected cost-to-go functions depend only on the stage rather than the nodes, i.e., we have $\mathcal{Q}_n(\cdot) \equiv \mathcal{Q}_t(\cdot)$ for all $n \in \mathcal{S}_t$. This allows for considerable reduction in the number of DP equations (1.3). By exploiting stage-wise independence, a sampling-based nested decomposition method – Stochastic Dual Dynamic Programming (SDDP) is proposed in [63]. This algorithm iterates between forward and backward steps. In the forward step, a subset of scenarios is sampled from the scenario tree and optimal solutions for each sample path are computed for each of them independently. Then in the backward step, starting from the last stage, the algorithm adds supporting hyperplanes to the approximate cost-to-go functions of the previous stage. These hyperplanes are Benders' cuts evaluated at the optimal solutions from the previous stage. After solving the problem at the first stage, a lower bound on the policy value can be obtained. It is then compared against a statistical upper bound computed from the forward step. Various proofs of almost sure convergence of SDDP under mild assumptions have been proposed [see e.g., 23; 67; 76; 32]. The SDDP algorithm has also been embedded in the scenario tree framework [68], and extended to risk averse multistage linear programming problems [76; 78].

Many multistage stochastic programming applications require integer variables for modeling complex constraints. For example, in stochastic generation scheduling problems, complex constraints such as minimum up and down times, and start-up and shut-down costs are modeled using binary variables. While enormous amount of work has been done in both theory and solution strategies for two-stage ($T = 2$) stochastic integer programs, the progress on multistage stochastic integer programming is somewhat limited [see e.g., 2; 70]. In MSIP, due to the presence of integer variables, the convexity and continuity of the future cost-to-go functions are lost. A natural way to tackle such problem is to consider the extensive form of the problem, and then relax the coupling constraints so that it can be decomposed into scenario-based or component-based subproblems. Different decomposition algorithms involving dual decomposition such as progressive hedging algorithm [69; 81; 31], scenario decomposition and Lagrangian relaxation [19; 58; 24], and multistage cluster Lagrangian and primal decompositions [28; 15; 72; 85] have been successful in solving various classes of MSIP problems. MSIP problems with binary state variables are studied in [6], and a branch-and-fix coordination approach is proposed, which coordinates the selection of the branching nodes and branches variables in the scenario subproblems such that will be jointly optimized. All of the above approaches are based on the extensive form (1.1) of MSIP or explicitly deal with the entire scenario tree, and do not scale well to large scenario trees.

Existing attempts at extending the nested decomposition and SDDP approaches for the dynamic programming formulation (1.2)-(1.3) for MSIP and other nonconvex problem are based on convex relaxations of the cost-to-go functions. For example, relaxing the integrality constraints so that the problem becomes an MSLP problem [57; 29; 50]; combining stochastic dynamic programming and SDDP methods to retain the convexity [33; 40]. Another way of dealing with non-convexity is to approximate the cost-to-go functions directly. For instance, approximating the bilinear relationship between variables using McCormick envelops is studied in [21]. This approach is further improved by optimizing the Lagrangian multipliers, which results in tighter cuts [83]. More recently, the concept of locally valid cuts is introduced and integrated in the SDDP framework [1]. Note that all the above methods produce solutions to different forms of relaxations rather than the original problem. In [65], authors propose a new extension of SDDP, which, rather than cutting planes, uses step functions to approximate the value function.

1.3 Contributions

As noted above, the nonconvexity of integer programming value functions makes it impossible to directly adapt nested decomposition algorithms such as Benders' decomposition and its stochastic variant, SDDP, to MSIP. In this paper, we propose a stochastic dual dynamic integer programming algorithm for solving MSIP with *binary* state variables. The key contributions are summarized below.

1. We propose a stochastic nested decomposition (SND) algorithm and its practical realization, namely the Stochastic Dual Dynamic integer Programming (SDDiP) algorithm when stochasticity satisfies stage-wise independence, to solve general MSIP problems with binary state variables. We define a precise notion of valid, tight, and finite cuts, and provide a rigorous proof of the finite convergence with

probability one of the SND, therefore SDDiP, to an optimal policy if the cuts satisfy these three conditions and sampling is done with replacement. The proposed algorithms provide a general framework of solving MSIP problems to optimality and redirects the question to constructing valid and tight cuts for nonconvex expected cost-to-go functions at each node.

2. We propose a new class of cutting planes, called Lagrangian cuts, by considering a reformulation of the nodal subproblems and solving its Lagrangian dual problem. In such a reformulation, we make local copies of the state variables, and the corresponding constraints are relaxed in the Lagrangian dual. We prove that these cuts satisfy our proposed notion of valid and tight cuts by showing strong duality holds for the Lagrangian dual. A simplified version of a Lagrangian cut strengthens the usual Benders' cut. While strengthened Benders' cuts are not necessarily tight, our computational experience indicates that they provide significant benefits.

3. Extensive numerical tests are presented to demonstrate the effectiveness of the SDDiP algorithm. In particular, we apply SDDiP with different combination of cutting planes to three classes of large-scale MSIP problems that have practical importance: a power generation capacity planning problem, a multistage portfolio optimization problem, and an airline revenue management problem. A particularly notable feature is that we transform non-binary state variables in these problems, either integer or continuous, to binary state variables. The promising results demonstrate the applicability of SDDiP for solving MSIP with *general* (i.e. not necessarily binary) state variables.

This paper is organized as follows. In Section 2, we describe the class of MSIP problem we consider in this work and propose a key reformulation. In Section 3, we present the SND and SDDiP algorithms and prove their finite convergence with probability one with valid, tight, and finite cuts. Section 4 contains the development of Lagrangian cuts as well as the proof of its validity and tightness. Numerical experiments together with discussions are included in Section 6. Finally, we provide some concluding remarks in Section 7.

2 MSIP with Binary State Variables

We consider multistage stochastic mixed integer linear programming problems, i.e., we make the following assumptions regarding the MSIP (1.1)

(A1) The objective function $f_n(x_n, y_n)$ in each node n is a linear function in x_n and y_n, and the constraint set X_n is a nonempty compact mixed integer polyhedral set.

The results in this paper can be easily extended to settings with nonlinear objective functions and constraint sets under mild regularity conditions. However, to make the main idea clear, we focus on the linear case.

A key requirement of our developments is that the state variables x_n in (1.1) are binary. The local variables y_n, however, can be general mixed integer. Recall that, in the presence of integer local variables, the value functions and expected cost-to-go functions are nonconvex with respect to the state variables. Existing nested decomposition algorithms use piece-wise convex polyhedral representations of these functions. In general, it is impossible to construct such convex polyhedral representations of the nonconvex value functions that are tight at the evaluated state variable values. On the other hand, any function of binary variables can be represented as a convex polyhedral function. We exploit this fact to develop exact nested decomposition algorithms for MSIP with binary state variables. Moreover, as discussed in Section 5, any MSIP with mixed integer state variables can be approximated to desired precision with an MSIP with binary state variables without increasing the problem size by too much. Thus the proposed SDDiP can be used to approximately solve very large class of MSIP problems. This is substantiated by our computational results.

Definition 1. We say that an MSIP of the form (1.1) has *complete continuous recourse* if, for any value of the state variables and the local integer variables, there exist values for the continuous local variables such that the resulting solution is feasible. That is, suppose $y_n = (u_n, v_n)$ where $u_n \in \mathbb{Z}_+^{\ell_1}$ and $v_n \in \mathbb{R}_+^{\ell_2}$, then given any $(\hat{x}_{a(n)}, \hat{x}_n, \hat{u}_n)$, there exists $\hat{v}_n \in \mathbb{R}_+^{\ell_2}$ such that $(\hat{x}_{a(n)}, \hat{x}_n, (\hat{u}_n, \hat{v}_n)) \in X_n$ for all $n \in \mathcal{T}$.

In addition to (A1) we also make the following assumption

(A2) Problem (1.1) has *complete continuous recourse*.

The above assumption can always be achieved by adding nonnegative auxiliary continuous variables and penalizing them in the objective function.

Next, we introduce a simple, but key reformulation of (1.1) based on making local copies of the state variables. That is, we introduce an auxiliary variable z_n for each node n and equate it to the parent node's state $x_{a(n)}$. The resulting formulation, which we consider for the remainder of this paper, is

$$
\begin{aligned}
\min_{x_n, y_n, z_n} \quad & \sum_{n \in \mathcal{T}} p_n f_n(x_n, y_n) \\
\text{s.t.} \quad & (z_n, x_n, y_n) \in X_n \quad \forall\, n \in \mathcal{T} && \text{(2.1a)} \\
& z_n = x_{a(n)} \qquad\quad\;\; \forall\, n \in \mathcal{T} && \text{(2.1b)} \\
& z_n \in [0,1]^d \qquad\quad \forall\, n \in \mathcal{T} && \text{(2.1c)} \\
& x_n \in \{0,1\}^d \qquad\quad \forall\, n \in \mathcal{T}. && \text{(2.1d)}
\end{aligned}
$$

This reformulation turns out to be crucial for the development of a class of valid and tight inequalities to approximate the cost-to-go functions. Detailed study of (2.1), especially a certain strong duality property, will be given in Section 4.3. The important role of the redundant constraint (2.1c) will become clear there. However, except in Section 4.3, we will fold constraint (2.1c) into X_n to save space.

Now we can write down the DP equations for the optimal value function of the multistage problem (2.1) at node $n \in \mathcal{T}$ as follows:

$$
\begin{aligned}
(P_1): \quad \min_{x_1, y_1, z_1} \quad & f_1(x_1, y_1) + \sum_{m \in \mathcal{C}(1)} q_{1m} Q_m(x_1) && \text{(2.2)} \\
\text{s.t.} \quad & (z_1, x_1, y_1) \in X_1 \\
& z_1 = x_{a(1)} \\
& x_1 \in \{0,1\}^d.
\end{aligned}
$$

where for each node $n \in \mathcal{T} \setminus \{1\}$,

$$
\begin{aligned}
(P_n): \quad Q_n(x_{a(n)}) := \min_{x_n, y_n, z_n} \quad & f_n(x_n, y_n) + \sum_{m \in \mathcal{C}(n)} q_{nm} Q_m(x_n) && \text{(2.3)} \\
\text{s.t.} \quad & (z_n, x_n, y_n) \in X_n \\
& z_n = x_{a(n)} \\
& x_n \in \{0,1\}^d.
\end{aligned}
$$

3 Stochastic Nested Decomposition and SDDiP

In this section, we present a Stochastic Nested Decomposition (SND) algorithm and its special case, SDDiP, when the stochasticity satisfies stage-wise independence, for solving the MSIP (2.1) with binary state variables. The proposed SND and SDDiP algorithms solve the DP recursion (2.3) by sampling the scenario tree and iteratively strengthening a convex piece-wise polyhedral lower approximation of the expected cost-to-go function $\mathcal{Q}_n(\cdot)$ at each node $n \in \mathcal{T}$. The key to the convergence of the SND, and therefore SDDiP, lies in a certain notion of tightness of the lower approximation of the value functions achieved by valid linear inequalities, which we will precisely define. In the following, we will first outline the SND algorithm, and then introduce the sufficient cut conditions, and prove the finite convergence with probability one of the SND algorithm to a global optimal solution of problem (2.1) under these conditions. Then, we will introduce the SDDiP algorithm, which provides a practical solution for solving MSIP with enormous scenario trees.

3.1 The SND Algorithm

The proposed SND algorithm is given in Algorithm 1. Details can be outlined as follows. In each iteration i, the SND algorithm consists of a sampling step, a forward step, and a backward step.

In the sampling step, a subset of scenarios, i.e., a set of paths from root to a subset of leaf nodes, is sampled from the tree. In particular, we consider the following sampling procedure: out of all the N nodes in the last stage of the scenario tree, M nodes, denoted as $\{n^i_{j_1}, \dots, n^i_{j_M}\}$, are sampled based on the distribution $\{p_n : n \in \mathcal{S}_T\}$. Let $\mathcal{P}^i(n_{j_k})$ denote the scenario path from root to the leaf node $n^i_{j_k}$. The set $\{\omega^i_k := \mathcal{P}^i(n_{j_k})\}_k$ contains all the corresponding scenario paths for all $k = 1, \dots, M$. The sampling can be done with or without replacement, and there is no significant practical difference between them as M is usually much smaller than N.

In iteration i, the forward step proceeds stage-wise from $t = 1$ to T by solving a DP equation with an approximate expected cost-to-go function at each sampled node $n \in \omega^i_k$. In particular, at node n with the parent node's state $x^i_{a(n)}$, the DP recursion (2.3) is approximated by the following forward problem

$$
\begin{aligned}
(P^i_n(x^i_{a(n)}, \psi^i_n)) : \quad \underline{Q}^i_n(x^i_{a(n)}, \psi^i_n) := \min_{x_n, y_n, z_n} \quad & f_n(x_n, y_n) + \psi^i_n(x_n) && (3.1a)\\
\text{s.t.} \quad & (z_n, x_n, y_n) \in X_n && (3.1b)\\
& z_n = x^i_{a(n)} && (3.1c)\\
& x_n \in \{0,1\}^d, && (3.1d)
\end{aligned}
$$

where $\psi^i_n(\cdot)$ is defined as:

$$
\begin{aligned}
\psi^i_n(x_n) := \min \big\{\theta_n : \quad & \theta_n \geq L_n, && (3.2a)\\
& \theta_n \geq \sum_{m \in \mathcal{C}(n)} q_{nm}(v^\ell_m + (\pi^\ell_m)^\top x_n), \ \forall \ell = 1, \dots, i-1\big\}. && (3.2b)
\end{aligned}
$$

In other words, the forward problem in iteration i is characterized by $x^i_{a(n)}$, which is obtained from solving its parent node $a(n)$'s forward problem, as well as by $\psi^i_n(\cdot)$ defined by (3.2a)–(3.2b), which provides a piecewise-linear convex lower-approximation of the expected cost-to-go function $\mathcal{Q}_n(x_n)$. Here, we assume there is a lower bound L_n in (3.2a) to avoid unboundedness of the forward problem. An optimal solution of the state variable in $(P^i_n(x^i_{a(n)}, \psi^i_n))$, denoted as x^i_n, is passed on to the forward problems $(P^i_m(x^i_n, \psi^i_m))$ of its children nodes $m \in \mathcal{C}(n)$. In other words, the forward step updates the state variable solution x^i_n for each $n \in \mathcal{T}$.

When all the forward problems on the sampled paths are solved in iteration i, the backward step starts from the last stage T. The goal of the backward step is to update the lower approximation ψ^i_n for each sampled node $n \in \omega^i_k$. In particular, in a last-stage sampled node $n \in \mathcal{S}_T$, a suitable relaxation of the forward problem $(P^i_n(x^i_{a(n)}, \psi^i_n))$, denoted as (R^i_n), is solved, which produces a linear inequality that lower approximates the true value function $Q_n(x^i_{a(n)})$. Note that the last stage problem does not have a cost-to-go function, therefore $\psi^i_n \equiv 0$ for all i. Going back one stage, at a sampled node $n \in \mathcal{S}_{T-1}$, all the linear inequalities generated from n's children nodes are aggregated in the form of (3.2b) and added to update its lower approximation from $\psi^i_n(\cdot)$ to $\psi^{i+1}_n(\cdot)$. Then, a suitable relaxation of the updated problem $(P^i_n(x^i_{a(n)}, \psi^{i+1}_n))$ is solved in the backward step at node n. This generates a new linear inequality, which will be aggregated to its parent's node. The backward step continues in this way until it reaches back to the root node of the tree.

Since the linear cuts in (3.2a)-(3.2b) are under-approximations of the true expected cost-to-go function, the optimal value of the forward problem (P^i_1) at node 1 provides a *lower* bound, LB, to the true optimal value of (2.1). However, it is important to note that the upper bound, UB, obtained by SND is only a statistical upper bound. Its validity is guaranteed with certain probability provided that M is not too small (e.g., $M > 30$). However, no matter how large M is, it could still happen that this upper bound is smaller than the valid lower bound evaluated in the backward step. As a result, one needs to be careful when using the stopping criterion $UB - LB \leq \epsilon$. Other stopping criteria are also used in the literature, e.g., stop the algorithm when the lower bounds become stable and the statistical upper bound given by a large sample size is close to the lower bound; or enforce a limit on the total number of iterations [78; 17].

Algorithm 1 :: Stochastic Nested Decomposition

Initialize: $LB \leftarrow -\infty$, $UB \leftarrow +\infty$, $i \leftarrow 1$, and an initial lower approximation $\{\psi_n^1(\cdot)\}_{n \in \mathcal{T}}$
while some stopping criterion is not satisfied **do**
  Sample M scenarios $\Omega^i = \{\omega_1^i, \ldots, \omega_M^i\}$

  /* Forward step */
  for $k = 1, \ldots, M$ **do**
    for $n \in \omega_k^i$ **do**
      solve forward problem $P_n^i(x_{a(n)}^i, \psi_n^i)$
      collect solution $(x_n^i, y_n^i, z_n^i, \theta_n^i = \psi_n^i(x_n^i))$
    end for
    $u^k \leftarrow \sum_{n \in \omega_k^i} f_n(x_n^i, y_n^i)$
  end for

  /* (Statistical) upper bound update */
  $\hat{\mu} \leftarrow \frac{1}{M} \sum_{k=1}^M u^k$ and $\hat{\sigma}^2 \leftarrow \frac{1}{M-1} \sum_{k=1}^M (u^k - \hat{\mu})^2$
  $UB \leftarrow \hat{\mu} + z_{\alpha/2} \frac{\hat{\sigma}}{\sqrt{M}}$

  /* Backward step */
  for $t = T-1, \ldots, 1$ **do**
    for $n \in \mathcal{S}_t$ **do**
      if $n \in \omega_k^i$ for some k **then**
        for $m \in \mathcal{C}(n)$ **do**
          solve a suitable relaxation (R_n^i) of the updated problem $P_n^i(x_{a(n)}^i, \psi_n^{i+1})$ and collect cut coefficients (v_m^i, π_m^i)
        end for
        add cut (3.2b) using the coefficients $\{(v_m^i, \pi_m^i)\}_{m \in \mathcal{C}(n)}$ to ψ_n^i to get ψ_n^{i+1}
      else
        $\psi_n^{i+1} \leftarrow \psi_n^i$
      end if
    end for
  end for

  /* Lower bound update */
  solve $P_1^i(\bar{x}_0, \psi_1^{i+1})$ and set LB be the optimal value
  $i \leftarrow i + 1$
end while

3.2 The SDDiP Algorithm

We now propose the SDDiP algorithm for the setting where the scenario tree satisfies stage-wise independence, i.e., for any two nodes n and n' in $\mathcal{S}_t$ the set of children nodes $\mathcal{C}(n)$ and $\mathcal{C}(n')$ are defined by identical data and conditional probabilities. In this case, the value functions and expected cost-to-go functions depend only on the stage rather than the nodes, i.e., we have $\mathcal{Q}_n(\cdot) \equiv \mathcal{Q}_t(\cdot)$ for all $n \in \mathcal{S}_t$. As a result, only one problem is maintained per stage, and cuts generated from different candidate solutions are added to the same problem.

We consider the setting where the scenario tree is created by sampling a stage-wise independent stochastic process. Let N_t be the number of realizations of uncertain parameters at stage t, each outcome has an equal probability of $1/N_t$. The total number of scenarios is $N = \prod_{t=1}^{T} N_t$. For any $1 \leq t \leq T$ and $i \geq 1$, let $\psi_t^i(\cdot)$ be the approximate expected cost-to-go function in stage t at the beginning of iteration i (cf. (3.1)-(3.2)). For a particular uncertain data realization $\xi_t^k (1 \leq k \leq N_t)$ in stage t, let $(P_t^i(x_{t-1}^{ik}, \psi_t^i, \xi_t^k))$ be the corresponding stage problem given state variable x_{t-1}^{ik} at the beginning of iteration i, and denote its optimal solution by $(x_t^{ik}, y_t^{ik}, z_t^{ik}, \theta_t^{ik})$. In the backward step, given a candidate solution x_{t-1}^{ik}, let (R_t^{ik}) be a suitable relaxation of the updated problem $(P_t^i(x_{t-1}^{ik}, \psi_t^{i+1}, \xi_t^j))$ for some $1 \leq j \leq N_t$, and (v_t^{ij}, π_t^{ij}) be the corresponding cut coefficients collected from solving the relaxation problem. Since each outcome of the uncertain data process has the same probability, the cut (3.2b) is obtained by taking the average of all generated cut coefficients, i.e.,

$$\theta_{t-1} \geq \frac{1}{N_t} \sum_{j=1}^{N_t} (v_t^{ij} + (\pi_t^{ij})^\top x_{t-1}). \tag{3.3}$$

The SDDiP algorithm is described in Algorithm 2, and its almost sure convergence immediately follows from Theorem 1.

Algorithm 2 :: Stochastic Dual Dynamic Integer Programming

```
Initialize: LB ← −∞, UB ← +∞, i ← 1, and an initial lower approximation {ψ_t^1(·)}_{t=1,...,T}
while some stopping criterion is not satisfied do
  Sample M scenarios Ω^i = {ξ_1^k, ..., ξ_T^k}_{k=1...,M}

  /* Forward step */
  for k = 1, ..., M do
    for t = 1, ..., T do
      solve forward problem P_t^i(x_{t-1}^{ik}, ψ_t^i, ξ_t^k)
      collect solution (x_t^{ik}, y_t^{ik}, z_t^{ik}, θ_t^{ik} = ψ_t^i(x_t^{ik}))
    end for
    u^k ← Σ_{t=1,...,T} f_t(x_t^{ik}, y_t^{ik}, ξ_t^k)
  end for

  /* (Statistical) upper bound update */
  μ̂ ← (1/M) Σ_{k=1}^M u^k and σ̂^2 ← (1/(M−1)) Σ_{k=1}^M (u^k − μ̂)^2
  UB ← μ̂ + z_{α/2} σ̂/√M

  /* Backward step */
  for t = T, ..., 2 do
    for k = 1, ..., M do
      for j = 1, ..., N_t do
        solve a suitable relaxation (R_t^{ij}) of the updated problem
        P_t^i(x_{t-1}^{ik}, ψ_n^{i+1}, ξ_t^j) and collect cut coefficients (v_t^{ij}, π_t^{ij})
      end for
      add cut (3.3) to ψ_{t-1}^i to get ψ_{t-1}^{i+1}
    end for
  end for

  /* Lower bound update */
  solve P_1^i(x̄_0, ψ_1^{i+1}) and set LB to the optimal value
  i ← i + 1
end while
```

For the problem with right hand side uncertainty, simple stage-wise dependency, e.g., p-th order autoregressive model, can be transformed into the independent case by adding additional decision variables [78]. However this approach in general does not extend to the situation where uncertainty exists in the objective coefficients or left hand side matrix of constraints because bilinear terms will be introduced but cannot be handled by the standard SDDP method. In our setting, however, these bilinear terms are products of two binary variables after reformulation using binary expansion or approximation, which can be easily reformulated as linear constraints. This is another significant advantage of considering the 0-1 state space.

3.3 Sufficient Cut Conditions

The SND and SDDiP algorithms have different implementations according to how the relaxation problem (R_n^i) is formed and how the cut coefficients are obtained in the backward step. However, regardless of detailed mechanisms for relaxation and cut generation, the SND and SDDiP algorithms are valid as long as the cuts satisfy the following three sufficient conditions, namely, they are *valid*, *tight*, and *finite*, as defined below.

Definition 2. Let $\{(v_n^i, \pi_n^i)\}_{n\in\Omega^i}$ be the cut coefficients obtained from the backward step of the i-th iteration of the SND or SDDiP algorithm. We say such a collection of cuts is

(i) *valid*, if for all $n \in \Omega^i$ and all iteration i,

$$Q_n(x_{a(n)}) \geq v_n^i + (\pi_n^i)^\top x_{a(n)} \quad \forall\, x_{a(n)} \in \{0,1\}^d, \tag{3.4}$$

(ii) *tight*, if for all $n \in \Omega^i$ and all iteration i,

$$\underline{Q}_n^i(x_{a(n)}^i, \psi_n^{i+1}) = v_n^i + (\pi_n^i)^\top x_{a(n)}^i, \tag{3.5}$$

where $\underline{Q}_n^i(x_{a(n)}^i, \psi_n^{i+1})$ is defined in (3.1) and $x_{a(n)}^i$ is the solution of state variable $x_{a(n)}$ obtained from the forward step in iteration i, and

(iii) *finite*, if in each iteration i of the SND and SDDiP algorithms, solving the relaxation problem (R_n^i) of $(P_n^i(x_{a(n)}^i, \psi_n^{i+1}))$ can only generate finitely many different cut coefficients (v_n^i, π_n^i).

It is easy to see that valid cuts are needed. The tightness of the cuts means that the cut generated from solving a *relaxation* of $(P_n^i(x_{a(n)}^i, \psi_n^{i+1}))$ needs to exactly recover the objective value $\underline{Q}_n^i(x_{a(n)}^i, \psi_n^{i+1})$ of $(P_n^i(x_{a(n)}^i, \psi_n^{i+1}))$ at $x_{a(n)}^i$. The tightness property alludes to a certain strong duality of the cuts that we will introduce in Section 4.3, and is crucial in ensuring the convergence of the SND and SDDiP algorithms. The finiteness condition is important to guarantee finite convergence. In Section 4, we discuss various types of relaxations and associated cuts that can be used in the proposed algorithms. Before this, let us first prove the convergence of SND and SDDiP algorithms using the three proposed properties of cuts.

3.4 Convergence

In this section, we prove the convergence of the SND algorithm, the convergence result for SDDiP algorithm naturally follows. In particular, we show that, with probability one, the approximate cost-to-go functions constructed using valid, tight, and finite cuts define an optimal solution to MSIP with binary state variables in a finite number of iterations. We have the following technical assumption.

(A3) In any node $n \in \mathcal{T}$ and iteration i in the SND algorithm, given the same parent solution $x_{a(n)}^i$ and the same approximate cost-to-go function ψ_n^i, the nodal problem $P_n^i(x_{a(n)}^i, \psi_n^i)$ is always solved to the same optimal solution x_n^i.

This assumption is to avoid the situation, where the algorithm for solving the same nodal problem keeps generating different optimal solutions (if they exist). Most deterministic MIP solvers, e.g. CPLEX and Gurobi, satisfy (A3). Therefore, it is a practical assumption. However, we do not assume the nodal problem $P_n^i(\cdot)$ has a unique optimal solution.

Theorem 1. *Suppose the sampling procedure in the forward step is done with replacement, the cuts generated in the backward step are valid, tight, and finite, and the algorithm for solving the nodal problems $\{P_n^i(\cdot)\}_{n\in\mathcal{T}}$ satisfies (A3), then with probability one, the forward step of the SND algorithm defines an optimal solution to the multistage stochastic program (2.1) after a finite number of iterations.*

Proof. First, notice that each binary state variable x_n in (2.1) can only take at most 2^d different values and the cutting planes used in the backward steps are finite (see Definition 2), it follows that there are finitely many possible realizations (polyhedral models) for the approximate expected cost-to-go functions $\{\psi_n^i(\cdot)\}_{n\in\mathcal{T}}$ for all $i \geq 1$.

At the beginning of any iteration $i \geq 1$, the current approximate expected cost-to-go functions $\{\psi_n^i(\cdot)\}_{n\in\mathcal{T}}$ define a solution (x_n^i, y_n^i) over the tree obtained by the forward step of iteration i, i.e.,

$$(x_n^i, y_n^i) \in \operatorname{argmin} \left\{ \begin{array}{ll} \min\limits_{x_n, y_n} & f_n(x_n, y_n) + \psi_n^i(x_n) \\ \text{s.t.} & (x_{a(n)}^i, x_n, y_n) \in X_n \quad \forall n \in \mathcal{T} \end{array} \right\}. \tag{3.6}$$

It is worth noting that during a particular iteration, the SND algorithm does not compute all of these solutions but only those along the sampled paths (scenarios). We first prove the following claim, which gives a sufficient condition under which the solution defined in (3.6) is optimal to the original problem.

Claim 1. If, at iteration i of the SND algorithm, $\psi_n^i(x_n^i) = \mathcal{Q}_n(x_n^i)$ for all $n \in \mathcal{T}$, then the forward solution $\{x_n^i, y_n^i\}_{n\in\mathcal{T}}$ is optimal to problem (2.1).

Proof of Claim 1: Since the cuts generated in backward steps are valid, $\{\psi_n^i(\cdot)\}_{n\in\mathcal{T}}$ is a lower approximation to the true expected cost-to-go functions, i.e., $\psi_n^i(x_n) \leq \mathcal{Q}_n(x_n)$ for all $x_n \in \{0,1\}^d$ and $n \in \mathcal{T}$. Therefore, $\underline{Q}_n^i(x_{a(n)}^i, \psi_n^i) \leq Q_n(x_{a(n)}^i)$ (cf. (2.3) and (3.1)). Furthermore, we have

$$\underline{Q}_n^i(x_{a(n)}^i, \psi_n^i) = f_n(x_n^i, y_n^i) + \psi_n^i(x_n^i) \tag{3.7a}$$
$$= f_n(x_n^i, y_n^i) + \mathcal{Q}_n(x_n^i) \tag{3.7b}$$
$$\geq Q_n(x_{a(n)}^i), \tag{3.7c}$$

where (3.7a) is true because x_n^i by definition is an optimal solution of $(P_n^i(x_{a(n)}^i, \psi_n^i))$, (3.7b) follows the assumption $\psi_n^i(x_n^i) = \mathcal{Q}_n(x_n^i)$, and (3.7c) holds because (x_n^i, y_n^i) is feasible for the true DP recursion (2.3). Therefore, (x_n^i, y_n^i) is also optimal for the true DP recursion (2.3) for all $n \in \mathcal{T}$, thus (x_n^i, y_n^i) is optimal for (2.1). This completes the proof of Claim 1. $\diamond$

Suppose the solution defined by (3.6) at the beginning of iteration i is not optimal, then there must exist some $n \in \mathcal{T}$ such that $\psi_n^i(x_n^i) < \mathcal{Q}_n(x_n^i)$. Any iteration $j \geq i$ can be characterized as either one of the following two types:

(a) $\{\psi_n^{j+1}(\cdot)\}_{n\in\mathcal{T}} \neq \{\psi_n^j(\cdot)\}_{n\in\mathcal{T}}$, i.e., at least one $\psi_n^j(\cdot)$ changes during the backward step;
(b) $\{\psi_n^{j+1}(\cdot)\}_{n\in\mathcal{T}} = \{\psi_n^j(\cdot)\}_{n\in\mathcal{T}}$, i.e., all $\psi_n^j(\cdot)$ remain the same after the backward step.

It is possible that consecutive iterations after i may belong to Type-a or Type-b iterations. Let us denote I_a^k and I_b^k as the k-th such set of consecutive Type-a and Type-b iterations, respectively. Let $K = \sup\{i : \{x_n^i, y_n^i\}_{n\in\mathcal{T}}$ is not optimal$\}$, and let K_a and K_b respectively be the total number of sets of consecutive Type-a and Type-b iterations, when the forward tree solution $\{x_n^i, y_n^i\}_{n\in\mathcal{T}}$ is not optimal. Let us also denote $|I_a^k|$ and $|I_b^k|$ as the cardinality of the k-th set of consecutive Type-a and Type-b iterations, respectively. Since there are only finitely many cuts that can be added, both K_a and each $|I_a^k|$ must be finite. As will be shown below, each I_b^k occurrence before the SND algorithm converges is followed by a Type-a iteration. Therefore, $K_b \leq K_a$, hence K_b is also finite. We next show that each $|I_b^k|$ is finite with probability 1.

Claim 2. With probability 1, $|I_b^k|$ is finite for all $1 \leq k \leq K_b$.

Proof of Claim 2: For any $1 \leq k \leq K_b$, let j_k be the iteration when I_b^k starts, since $\{\psi_n^{j_k+1}(\cdot)\}_{n\in\mathcal{T}} = \{\psi_n^{j_k}(\cdot)\}_{n\in\mathcal{T}}$ and by assumption (A3), we have $\{x_n^{j_k+1}, y_n^{j_k+1}\}_{n\in\mathcal{T}} = \{x_n^{j_k}, y_n^{j_k}\}_{n\in\mathcal{T}}$. Because the solution $\{x_n^{j_k}, y_n^{j_k}\}_{n\in\mathcal{T}}$ is not optimal, by Claim 1, there exists $n_{j_k} \in \mathcal{T}$ such that $\psi_{n_{j_k}}^{j_k}(x_{n_{j_k}}^{j_k}) < \mathcal{Q}_{n_{j_k}}(x_{n_{j_k}}^{j_k})$. Choose such an node

n_{j_k} so that $t(n_{j_k})$ is the largest, hence for all $m \in \mathcal{C}(n_{j_k})$, $\psi_m^{j_k}(x_m^{j_k}) = \mathcal{Q}_m(x_m^{j_k})$. The sampling in the forward step is done with replacement, thus each scenario is sampled independently. Since there are finitely many scenarios, and each one is sampled with a positive probability, we know that with probability 1, after finitely many number of iterations, a scenario that contains node n_{j_k} will be sampled in an iteration, say j_k'. In the backward step of iteration j_k', the same state vector $x_{n_{j_k}}^{j_k'} = x_{n_{j_k}}^{j_k}$ will be evaluated at all children nodes of n_{j_k}, and a cut will be added to $\psi_{n_{j_k}}^{j_k'}(\cdot)$. We want to show that $\psi_{n_{j_k}}^{j_k'+1}(x_{n_{j_k}}^{j_k}) = \mathcal{Q}_{n_{j_k}}(x_{n_{j_k}}^{j_k})$ after adding this cut. Note that we have the following relations:

$$\psi_{n_{j_k}}^{j_k'+1}(x_{n_{j_k}}^{j_k}) \geq \sum_{m \in \mathcal{C}(n_{j_k})} q_{n_{j_k}m}(v_m^{j_k} + (\pi_m^{j_k})^\top x_{n_{j_k}}^{j_k}) \tag{3.8a}$$

$$= \sum_{m \in \mathcal{C}(n_{j_k})} q_{n_{j_k}m}\underline{Q}_m^{j_k}(x_{n_{j_k}}^{j_k}, \psi_m^{j_k}) \tag{3.8b}$$

$$= \sum_{m \in \mathcal{C}(n_{j_k})} q_{n_{j_k}m}(f_m(x_m^{j_k}, y_m^{j_k}) + \psi_m^{j_k}(x_m^{j_k})) \tag{3.8c}$$

$$= \sum_{m \in \mathcal{C}(n_{j_k})} q_{n_{j_k}m}(f_m(x_m^{j_k}, y_m^{j_k}) + \mathcal{Q}_m^{j_k}(x_m^{j_k})) \tag{3.8d}$$

$$\geq \sum_{m \in \mathcal{C}(n_{j_k})} q_{n_{j_k}m}Q_m^{j_k}(x_{n_{j_k}}^{j_k}, \psi_m^{j_k'}) \tag{3.8e}$$

$$= \mathcal{Q}_{n_{j_k}}(x_{n_{j_k}}^{j_k}). \tag{3.8f}$$

The inequality in (3.8a) follows from the construction of $\psi_{n_{j_k}}^{j_k'+1}(x_{n_{j_k}}^{j_k})$ in (3.2). The equality in (3.8b) follows from the fact that $(v_m^{j_k}, \pi_m^{j_k})$ is a tight cut for the relaxation problem of $(P_m^{j_k}(x_{n_{j_k}}^{j_k}, \psi_m^{j_k+1}))$ and uses the definition of tight cuts given in (3.5). The equality in (3.8c) follows from the definition of $\underline{Q}_m^{j_k}$ in (3.1). The equality (3.8d) holds due to the fact for all $m \in \mathcal{C}(n_{j_k})$, $\psi_m^{j_k}(x_m^{j_k}) = \mathcal{Q}_m(x_m^{j_k})$. Then, (3.8e) follows because $(x_m^{j_k}, y_m^{j_k})$ is a feasible solution of the problem $(P_m^{j_k}(x_{n_{j_k}}^{j_k}, \psi_m^{j_k+1}))$ with the parent state $x_{n_{j_k}}^{j_k}$ as defined in (2.3). Lastly, (3.8f) is the definition of $\mathcal{Q}_{n_{j_k}}(x_{n_{j_k}}^{j_k})$.

Since $\psi_{n_{j_k}}^{j_k'+1}(x_{n_{j_k}}^{j_k}) = \mathcal{Q}_{n_{j_k}}(x_{n_{j_k}}^{j_k})$, a new Type-a occurrence starts from the j_k'-th iteration. In other words, when the SND algorithm has not converged, i.e. $(x_n^i, y_n^i)_{n \in \mathcal{T}}$ is not optimal, each consecutive Type-b occurrence is followed by a Type-a iteration. This proves $K_b \leq K_a$. Therefore, the number of iterations in I_b^k for $1 \leq k \leq K_b$ is finite with probability 1. $\diamond$

It follows from Claim 2 that the condition in Claim 1 will hold after $K = \sum_{k=1}^{K_a} |I_a^k| + \sum_{k=1}^{K_b} |I_b^k|$ iterations. We have the following relations.

$$1 \geq Pr\left(\sum_{k=1}^{K_a} |I_a^k| + \sum_{k=1}^{K_b} |I_b^k| < \infty\right) = Pr\left(\sum_{k=1}^{K_b} |I_b^k| < \infty\right) = Pr\left(|I_b^k| < \infty, \forall\, 1 \leq k \leq K_b\right) = 1,$$

where the first equality follows from the finiteness of $\sum_{k=1}^{K_a} |I_a^k|$ and the second is due to $K_b < \infty$ for sure, and the last follows from Claim 2. Hence $Pr(K < \infty) = 1$. Therefore, the SND algorithm converges to an optimal solution of problem (2.1) in a finite number of iterations with probability 1. $\square$

4 Cut families

In this section, we discuss various types of cuts that can be used within the proposed algorithms. We discuss the well known Benders' and integer optimality cuts, and introduce the *Lagrangian* cuts derived from a Lagrangian relaxation corresponding to the reformulation (2.1), where local copies of state variables are introduced, and an associated collection of *strengthened Benders'* cuts.

4.1 Benders' Cut

A well known family of cuts is the Benders' cut [11], where the relaxation (R_n^i) solved in the backward step is the LP relaxation of problem $(P_n^i(x_{a(n)}^i, \psi_n^{i+1}))$. Therefore, the cost coefficients (v_n^i, π_n^i) are computed based on the optimal value of the LP relaxation and a basic optimal dual solution. Specifically, the cut added to node n in the backward step evaluated at a forward solution x_n^i takes the following form

$$\theta_n \geq \sum_{m\in\mathcal{C}(n)} q_{nm} Q_m^{LP}(x_n^i) + \sum_{m\in\mathcal{C}n} q_{nm} (\pi_m^i)^\top (x_n - x_n^i), \tag{4.1}$$

where $Q_m^{LP}(x_n^i)$ is the optimal LP relaxation objective function value of problem $(P_m^i(x_n^i, \psi_m^{i+1}))$ and π_m^i is a basic optimal dual solution corresponding to constraints $z_m = x_n^i$. This is the cut family used in nested decomposition algorithms for MSLP. For MSIP, Benders' cut are valid and finite (when basic dual optimal solutions are used) but not tight in the sense of (3.5) in general. Accordingly, for MSIP, the SND and SDDiP algorithms are not guaranteed to produce an optimal solution using only Benders' cuts.

4.2 Integer Optimality Cut

Another interesting collection of cutting planes is introduced by [48] and is designed for solving two-stage stochastic programs with binary first-stage variables. It is generated by evaluating the subproblem at a feasible first-stage solution and coincides with the true expected cost-to-go function at the proposed first-stage solution. We present a natural extension of them to the SND and SDDiP algorithms for the multistage setting.

Let x_n^i be a solution to the problem $(P_n^i(x_{a(n)}^i \psi_n^i))$ solved in iteration i at node n in the forward step. The relaxations solved in the backward step are the original problems themselves. That is, let v_m^{i+1} be the optimal objective value of problem $(R_m^i) = (P_m^i(x_n^i, \psi_m^{i+1}))$ given x_n^i for all $m \in \mathcal{C}(n)$. Then the integer optimality cut added to $(P_n^i(x_{a(n)}^i, \psi_n^i))$ in the backward step takes the following form

$$\theta_n \geq (\bar{v}_n^{i+1} - L_n)\left(\sum_j (x_{n,j}^i - 1)x_{n,j} + \sum_j (x_{n,j} - 1)x_{n,j}^i\right) + \bar{v}_n^{i+1}, \tag{4.2}$$

where $\bar{v}_n^{i+1} = \sum_{m\in\mathcal{C}(n)} q_{nm} v_m^{i+1}$. It is easy to verify that integer optimality cuts are valid, tight and finite. Thus the proposed algorithms with this cut family is an exact approach for solving MSIP with binary state variables. However, these cuts are only tight at the proposed binary solution x_n^i and could be very loose at other solutions, and hence may not perform satisfactorily.

4.3 Lagrangian Cut

We consider another class of cuts obtained by solving a Lagrangian dual of the nodal forward problems. The relaxation solved in the backward step of iteration i in node n in this case is:

$$(R_n^i): \quad \max_{\pi_n} \left\{ \mathcal{L}_n^i(\pi_n) + \pi_n^\top x_{a(n)}^i \right\} \tag{4.3}$$

where

$$\begin{aligned} \mathcal{L}_n^i(\pi_n) \;=\; \min_{x_n, y_n, z_n, \theta_n} \quad & f_n(x_n, y_n) + \theta_n - \pi_n^\top z_n \\ \text{s.t.} \quad & (z_n, x_n, y_n) \in X_n \\ & x_n \in \{0,1\}^d \\ & z_n \in [0,1]^d \\ & \theta_n \geq L_n \\ & \theta_n \geq \sum_{m\in\mathcal{C}(n)} q_{nm}(v_m^\ell + (\pi_m^\ell)^\top x_n) \quad \forall \ell = 1, \ldots, i. \end{aligned} \tag{4.4}$$

We will denote the feasible region defined by the first four constraint systems of $\mathcal{L}_n^i(\pi_n)$ as X_n' and that defined by all five constraint systems as X_n''.

Given any $\{x_n^i\}_{n\in\Omega^i}$ with $x_n^i \in \{0,1\}^d$, a collection of cuts given by the coefficients $\{(v_n^i, \pi_n^i)\}_{n\in\Omega^i}$ is generated in the backward step of iteration i, where π_n^i is an optimal solution to the Lagrangian dual problem (R_n^i) and $v_n^i = \mathcal{L}_n^i(\pi_n^i)$ for all $n \in \Omega^i$. We call this collection of cuts the *Lagrangian cuts*.

Theorem 2. *Given any $\{x_n^i\}_{n\in\Omega^i}$ with $x_n^i \in \{0,1\}^d$, let π_n^i be an optimal solution to the Lagrangian dual problem (R_n^i) in (4.3) and $v_n^i = \mathcal{L}_n^i(\pi_n^i)$. Then, the collection of Lagrangian cuts $\{(v_n^i, \pi_n^i)\}_{n\in\Omega^i}$ is valid and tight in the sense of (3.4)-(3.5).*

Proof. First, we prove that the Lagrangian cuts generated in iteration i of the SND or SDDiP algorithm are tight at the forward solution $\{x_n^i\}_{n\in\Omega^i}$. The tightness of the Lagrangian cuts is essentially implied by a strong duality between the Lagrangian relaxation defined by (4.3)-(4.4) and the forward problem $(P_n^i(x_{a(n)}^i, \psi_n^i))$ defined in (3.1). Then, we prove by induction that they are also valid cuts.

Take any node $n \in \Omega^i$. Let π_n^i be an optimal dual solution of (4.3). Then, we have the following equalities:

$$\begin{aligned}\mathcal{L}_n^i(\pi_n^i) + (\pi_n^i)^\top x_{a(n)}^i &= \min\left\{f_n(x_n,y_n) + \theta_n - (\pi_n^i)^\top(z_n - x_{a(n)}^i) : (z_n,x_n,y_n,\theta_n) \in X_n''\right\}\\ &= \min\left\{f_n(x_n,y_n) + \theta_n : (z_n,x_n,y_n,\theta_n) \in \operatorname{conv}(X_n''), z_n = x_{a(n)}^i\right\},\end{aligned} \tag{4.5}$$

where (4.5) follows from Theorem 6.2 in [56]. Let $(\hat z_n, \hat x_n, \hat y_n, \hat\theta_n) \in \operatorname{conv}(X_n'')$ be an optimal solution of (4.5). Then there exists $\{(\hat z_n^k, \hat x_n^k, \hat y_n^k, \hat\theta_n^k)\}_{k\in K} \in X_n''$ such that $(\hat z_n, \hat x_n, \hat y_n, \hat\theta_n) = \sum_{k\in K}\lambda_k\cdot(\hat z_n^k, \hat x_n^k, \hat y_n^k, \hat\theta_n^k)$, where K is a finite set, $\lambda_k \ge 0$ for all $k \in K$, and $\sum_{k\in K}\lambda_k = 1$. Since $x_{a(n)}^i \in \{0,1\}^d$ and $\hat z_n^k \in [0,1]^d$ for all k, we have that $\sum_{k\in K}\lambda_k \hat z_n^k = \hat z_n = x_{a(n)}^i$, which implies that $\hat z_n^k = x_{a(n)}^i$ for all k. Thus $(\hat z_n, \hat x_n, \hat y_n, \hat\theta_n) \in \operatorname{conv}(X_n'' \wedge \{z_n = x_{a(n)}^i\})$ and

$$\begin{aligned}\mathcal{L}_n^i(\pi_n^i) + (\pi_n^i)^\top x_{a(n)}^i &= \min\left\{f_n(x_n,y_n) + \theta_n : (z_n,x_n,y_n,\theta_n) \in \operatorname{conv}(X_n'' \wedge \{z_n = x_{a(n)}^i\})\right\}\\ &= \min\left\{f_n(x_n,y_n) + \theta_n : (z_n,x_n,y_n,\theta_n) \in X_n'', z_n = x_{a(n)}^i\right\}\\ &= f_n(x_n^i, y_n^i) + \theta_n^i = \underline{Q}_n^i(x_{a(n)}^i, \psi_n^i),\end{aligned}$$

where the second equality follows since $f_n(x_n, y_n)$ is linear. This proves the tightness of the Lagrangian cuts according to (3.5).

Next, we show by induction that the Lagrangian cuts are valid. For the base case, we consider any sampled node $n \in \Omega^i$ and in the last stage $\mathcal{S}_T$. Note that $\psi_n^i \equiv 0$ in this last stage problem. Relaxing the constraint $z_n = x_{a(n)}$ in the definition (2.3) of $Q_n(x_{a(n)})$ using the optimal multiplier π_n^i of (4.3), we have for any $x_{a(n)} \in \{0,1\}^d$,

$$\begin{aligned}Q_n(x_{a(n)}) &\ge \min\{f_n(x_n,y_n) - (\pi_n^i)^\top(z_n - x_{a(n)}) : (z_n,x_n,y_n) \in X_n'\}\\ &= \min\{f_n(x_n,y_n) - (\pi_n^i)^\top z_n : (z_n,x_n,y_n) \in X_n'\} + (\pi_n^i)^\top x_{a(n)}\\ &= \mathcal{L}_n^i(\pi_n^i) + (\pi_n^i)^\top x_{a(n)}.\end{aligned}$$

Thus the Lagrangian cut is valid at any sampled $n \in \mathcal{S}_T$. For the induction step, consider a sampled node $n \in \mathcal{S}_t$ with $t \le T-1$, and assume that the Lagrangian cuts defined by $\{(v_m^i, \pi_m^i)\}_{m\in\mathcal{C}(n)}$ are valid. Note that

$$Q_n(x_{a(n)}) = \min\Big\{f_n(x_n,y_n) + \theta_n : (z_n,x_n,y_n) \in X_n,\ z_n = x_{a(n)},\ \theta_n \ge \sum_{m\in\mathcal{C}(n)} q_{nm}Q_m(x_n)\Big\}. \tag{4.6}$$

Since the cuts defined by $\{(\pi_m^i, v_m^i)\}_{m\in\mathcal{C}(n)}$ are valid, i.e. $Q_m(x_n) \ge v_m^i + (\pi_m^i)^\top x_n$ for any $x_n \in \{0,1\}^d$, X_n'' with these cuts is a relaxation of the feasible region of (4.6). Therefore, we have

$$Q_n(x_{a(n)}) \ge \min\left\{f_n(x_n,y_n) + \theta_n : (z_n,x_n,y_n,\theta_n) \in X_n'', z_n = x_{a(n)}\right\}$$

$$
\begin{aligned}
&\geq \min\left\{f_n(x_n,y_n)+\theta_n-(\pi_n^i)^\top z_n:(z_n,x_n,y_n,\theta_n)\in X_n''\right\}+(\pi_n^i)^\top x_{a(n)}\\
&=\mathcal{L}_n^i(\pi_n^i)+(\pi_n^i)^\top x_{a(n)},
\end{aligned}
$$

where the second inequality is by relaxing the constraint $z_n = x_{a(n)}$. Thus the Lagrangian cut defined by (π_n^i, v_n^i) is valid. This completes the proof of the theorem. □

If we restrict the set of dual optimal solutions π_n^i of (R_n^i) to be basic, then the set of Lagrangian cuts is also finite. Accordingly, the SND and SDDiP algorithms with this cut family are guaranteed to produced an optimal solution to MSIP with binary state variables in a finite number of iterations with probability one.

4.4 Strengthened Benders' Cut

The Lagrangian problem is an unconstrained optimization problem, thus for any fixed π_n, solving (4.4) to optimality yields a valid cut. Therefore, one can strengthen Benders' cut by solving a nodal mixed integer program. More concretely, we solve (4.4) at all $m \in \mathcal{C}(n)$ with π_m equal to a basic optimal LP dual solution π_m^i corresponding to the constraints $z_m = x_n^i$. Upon solving all these nodal subproblems, we can construct a valid cut which is parallel to the regular Benders' cut,

$$
\theta_n \geq \sum_{m\in\mathcal{C}(n)} q_{nm}\mathcal{L}_m(\pi_m^i)+\sum_{m\in\mathcal{C}n} q_{nm}(\pi_m^i)^\top x_n. \tag{4.7}
$$

Indeed, we have $\mathcal{L}_m(\pi_m^i) \geq Q_m^{LP}(x_n^i) - (\pi_m^i)^\top x_n^i$, thus (4.7) is at least as tight as Benders' cuts (4.1). For this reason, we call these cuts *strengthened Benders' cuts*. The strengthened Benders' cuts are valid and finite but are not guaranteed to be tight according to (3.5). Nonetheless these cuts afford significant computational benefits as demonstrated in Section 6.

Even though Lagrangian cuts are tight, whereas strengthened Benders' cuts are not in general, the latter are not necessarily dominated by the previous one, as shown in the following example.

Example 1. Consider the following two-stage program with only 1 scenario,

$$
\min_x \ \{x_1+x_2+Q(x_1,x_2): x_1,x_2\in\{0,1\}\}
$$

where $Q(x_1,x_2)=\min\{4y: y\geq 2.6-0.25x_1-0.5x_2, y\leq 4, y\in\mathbb{Z}_+\}$. It is easy to compute that $Q(0,0)=12$. The Benders' cut described in (4.1) is $\theta \geq 10 - x_1 - 2x_2$; the strengthened Benders' cut described in (4.7) is $\theta \geq 11 - x_1 - 2x_2$; and the Lagrangian cut is $\theta \geq 12 - 4x_2$. We see that the Lagrangian cut supports function $Q(x_1, x_2)$ at $(0,0)$, while the other two do not. Also, it is clear that the strengthened Benders' cut strictly improves the Benders' cut, and the strengthened Benders' cut and the Lagrangian cut do not dominate each other. ⋄

5 Dealing with General Mixed Integer State Variables

The development of SDDiP so far has been predicated by the assumption of binary state variables. Recall that, as discussed in Section 2, it is impossible to construct a convex polyhedral representation of a nonconvex value function using linear cuts that are tight at the evaluated state variable values unless they are binary. Thus, to apply the SDDiP approach to MSIP with general mixed integer state variables, we propose to approximate the state variables with their binary approximations. Under the assumptions of a bounded feasible region (assumption A1) and complete continuous recourse (assumption A2), such a binarization approach is justified by the following theorem.

Theorem 3. *For an MSIP with general mixed integer state variables satisfying assumptions (A1) and (A2) we can construct an approximating MSIP that has binary state variables such that any optimal solution to the approximating*

MSIP is an ε-optimal solution to the original MSIP, and the number, k, of the binary state variables per node in the approximating MSIP satisfies

$$k \leq d(\lfloor \log_2(M/\varepsilon) \rfloor + 1)$$

where d is the number of the state variables per node in the original MSIP and M is a positive constant depending on problem data.

Proof. See Appendix. □

Note that in many important applications the state variable dimension d is low, thus the above result indicates the resulting binary approximation is not too large since it scales only linearly with the d, and logarithmically with respect to the inverse of the precision required. Moreover, for applications where the state variables are general integer, we can set $\varepsilon = 1$.

A common criticism of binary reformulation of general integer variables is based on the classical paper [59]. In this work, the authors show that for mixed integer *linear* programs, binarizing *all* general integer variables is detrimental to the performance of MIP solvers on these problems. We contend that the conclusions from this work are not applicable to our setting. First, if we view an MSIP in its extensive form as a mixed integer linear program, then we binarize only a tiny fraction of the variables rather than all general integer variables as in [59]. In particular, for an MSIP with T stages, stage-wise independence, N nodes per stage, d state variables per stage, and n local variables per stage, the total number of variables in the extensive form is $(1 + N^{T-1})n + (T-1)d$ of which at most $(T-1)d$ state variables are binarized. Thus the fraction of binarized variables quickly approaches zero as the number of stages or the nodes per stage increase. On the other hand, if we view an MSIP as an optimization problem over the state variables only (by projecting out the local variables) as is done in the dynamic programming formulation, then the problem involves a nonlinear and nonconvex objective and again the conclusions of [59] for mixed integer linear programs are not applicable. In fact, recently a number of authors have demonstrated that binary approximations of continuous or general integer variables can be very effective for solving some classes of nonconvex nonlinear optimization problems (cf. [13; 37; 38]). The computational effectiveness of SDDiP on MSIP with general state variables after the proposed binary appoximation is demonstrated in the next section.

6 Computational Experiments

In this section, we present computational experiments to evaluate the SDDiP Algorithm 2 on a power generation expansion planning problem, a financial portfolio optimization problem, and an airline revenue management problem. Algorithm 2 is implemented in C++ with CPLEX 12.6.0 to solve the MIP and LP subproblems. The Lagrangian dual problem is solved to optimality using a basic subgradient algorithm [see e.g., 12, Sec. 6.3] with an optimality tolerance of 10^{-4}. All other relative MIP tolerance is set to 10^{-4} except when specified. All computations are conducted on a Linux (Fedora 22) desktop with four 2.4GHz processors and 8GB RAM.

6.1 Long-term Generation Expansion Planning

In a power generation expansion planning (GEP) problem, one seeks to determine a long-term construction and generation plan for different types of generators, taking into account the uncertainties in future demand and natural gas prices. Suppose there are n types of expansion technologies available. Let x_t be a vector representing numbers of different types of generators to be built in stage t, and y_t be a vector of the amount of electricity produced by each type of generator per hour in stage t. A deterministic formulation is as follows.

$$\min \quad \sum_{t=1}^{T} (a_t^\top x_t + b_t^\top y_t) \qquad \text{(investment cost + generation cost)}$$

$$\begin{aligned}
\text{s.t.} \quad & \forall\, t = 1, \dots, T \\
& \sum_{s=1}^{t} x_s \geq A_t y_t && \text{(generation capacity)} \\
& \sum_{s=1}^{t} x_s \leq \bar{u} && \text{(limitation on total number of generators)} \\
& \mathbf{1}^\top y_t = d_t && \text{(demand satisfaction)} \\
& x_t \in \mathbb{Z}_+^n,\ y_t \in \mathbb{R}_+^n.
\end{aligned}$$

In the above formulation, a_t and b_t are investment and generation cost at stage t, respectively. Matrix A_t contains maximum rating and maximum capacity information of generators, $\bar{u}$ is a pre-determined construction limits on each type of generators due to resource and regulatory constraints, and d_t is the electricity demand at stage t.

Scenario generation Among all data, $\{b_t\}_{t=1,\dots,T}$ and $\{d_t\}_{t=1,\dots,T}$ are subject to uncertainty. All data (except demand and natural gas price) used in this numerical study can be found in [45], where demand and natural gas price are modeled as two correlated geometric Brownian motions. We simplify the stochastic processes of electricity demand and natural gas price as follows. We assume that both processes are stage-wise independent. At each stage, electricity demand follows a uniform distribution, and natural gas price follows a truncated normal distribution with known first and second moments. In addition, these two processes are considered as independent to each other. There are six types of generators available for capacity expansion, namely Coal, Combined Cycle (CC), Combined Turbine (CT), Nuclear, Wind, and Integrated Gasification Combined Cycle (IGCC). Among these six types of generators, both CC and CT power generators are fueled by natural gas.

In the implementation, we create a new set of general integer variables s_t, representing the cumulative numbers of different types of generators built until stage t. After binary expansion, there are 48 binary state variables per stage. The local variables are x_t and y_t, containing 6 general integer variables and 7 continuous variables, respectively.

Performance Comparison We first consider an instance of the GEP problem with 10 decision stages. At each stage, three realizations of the uncertainty parameters are drawn, thus in total there are $3^9 = 19683$ scenarios with equal probability. We construct the extensive formulation on the scenario tree and use CPLEX to solve the problem as one large MIP. This formulation contains nearly 620,000 binary variables and 207,000 continuous variables. CPLEX returns an incumbent solution with an objective function value 7056.7, and the best bound 6551.6, i.e., a 7.16% gap remains after two hours.

We solve the same instance using SDDiP algorithm with seven different combinations of cutting planes and compare their performance. Each of the combinations includes at least one collection of tight cuts. The stopping criterion used in this numerical test is to terminate the algorithm once lower bounds obtained in the backward steps become stable, and the computation time limit is set to be 5 hours. After the lower bounds become stable, we evaluate the objective function value for 1500 forward paths independently, and construct a 95% confidence interval. The right endpoint of this interval is reported as the statistical upper bound of the optimal value. The seven combinations of cuts are specified below:

(1) Integer optimality cut (I);

(2) Lagrangian cut (L);

(3) Benders' cut + Integer optimality cut (B + I);

(4) Benders' cut + Lagrangian cut (B + L);

(5) Strengthened Benders' cut + Integer optimality cut (SB + I);

(6) Strengthened Benders' cut + Lagrangian cut (SB + L);

(7) Strengthened Benders' cut + Integer optimality cut + Lagrangian cut (SB + I + L).

In Table 1, we compare the performance of the SDDiP algorithm with integer optimality cuts (I) and Lagrangian cuts (L). The first column indicates the type of cuts; Column 2 represents the number of forward path sampled in the forward step; Column 3 contains the best lower bound computed by the algorithm when stopping criterion (or computation time limit) is reached; Column 4 shows the average number of iterations used; Column 5 contains a 95%-confidence statistical upper bound on the optimal value; Column 6 shows the gap between the statistical upper bound and the best lower bound in Column 2; and the last two columns contain the average total computation time and time used per iteration for each experiment setting.

Table 1: Performance of SDDiP algorithm with a single class of cutting planes

cuts	# FW	best LB	# iter	stat. UB	gap	time (sec.)	time/iter.
I	1	4261.1	4041	8999.1	52.65%	18000	4.5
	2	4184.5	2849	9005.5	53.53%	18000	6.3
	3	4116.2	2426	10829.9	61.99%	18000	7.4
	5	3970.4	1908	9730.0	59.19%	18000	9.4
	10	3719.8	1384	9868.5	62.31%	18000	13.0
	20	3427.8	969	10011.1	65.76%	18000	18.6
	50	3055.8	603	10002.9	69.45%	18000	29.9
L	1	6701.1	110	6762.4	0.91%	1810	16.5
	2	6701.1	57	6781.9	1.19%	1021	18.0
	3	6701.0	45	6769.5	1.01%	1595	35.5
	5	6701.1	36	6851.8	2.20%	741	20.6
	10	6701.3	34	6796.6	1.40%	1223	36.0
	20	6701.2	28	6803.3	1.50%	1274	45.5
	50	6701.1	30	6801.6	1.48%	2092	69.7

From Table 1 we can see that, if only integer optimality cuts are used in the backward step, the lower bound improves very slowly. As a result, it takes a long time for the algorithm to stop. In fact, none of the experiments converges within 5 hours of computation time and large gaps are observed between the lower and upper bounds on the optimal values. In comparison, if only Lagrangian cuts are used, the algorithm converges much faster. The lower bounds obtained are also significantly higher than those attained only with integer optimality cuts. In addition, for the Lagrangian cuts, the gap between the statistical upper bound and the deterministic lower bound is very small in all experiments with different choices of the number of forward sample paths. The reason behind these results should be clear from the construction of integer optimality cuts. Namely, they are much looser than Lagrangian cuts everywhere else except at the candidate solution being evaluated.

Table 2 presents similar computational results but in addition to using a single class of tight cuts (i.e. I or L), we further adopt either Benders' cuts or strengthened Benders' cuts (i.e. B or SB). We have the following comparisons.

1. *(B+I) v.s. I*: It is observed that adding Benders' cuts together with integer optimality cuts (B+I) leads to a significant improvement of the algorithm performance, comparing to the performance of only integer optimality cuts (I) in Table 1. Not only all experiments converge within 5 hours, the quality of the solutions is also very satisfactory, i.e., the gap between the statistical upper bound and deterministic lower bound is small ($\leq 2\%$) in most cases.
2. *(B+I) v.s. (SB+I) and (B+L)*: Another significant improvement on the algorithm performance can be observed by comparing (B + I) and (SB + I) of Table 2, where we substitute Benders' cuts with strengthened Benders' cuts. We can still attain small gaps, i.e., good estimations on the optimal value. Moreover, the number of iterations, the total time, and the average computation time all significantly decrease due to the tighter strengthened Benders' cuts. Comparing (B + I) with (B + L) suggests that replacing integer optimality cuts with Lagrangian cuts also results in a major improvement in both the total number of iterations and computation time.

3. *(SB+I) v.s. (SB+L) and (SB+I+L)*: No significant improvement is observed between (SB + I) and (SB + L). This is because the optimal Lagrangian dual multipliers do not deviate much from the LP dual optimal in these instances. Therefore, strengthened Benders' cuts and Lagrangian cuts are "similar" in this sense. Finally, adding integer optimality cuts in addition to the strengthened Benders' and Lagrangian cuts (SB + I + L) does not significantly affect algorithm performance, because integer optimality cuts do not contribute much in approximating the expected cost-to-go functions except at the candidate solutions.

Table 2: Performance of SDDiP algorithm with multiple classes of cutting planes

cuts	# FW	best LB ($MM)	# iter	stat. UB ($MM)	gap	time (sec.) (sec.)	time/iter. (sec.)
B + I	1	6701.1	399	6874.7	2.53%	3905	9.8
	2	6701.1	263	6757.1	0.83%	3524	13.4
	3	6701.0	204	6755.8	0.81%	3594	17.6
	5	6701.1	173	6799.5	1.44%	4457	25.8
	10	6701.1	146	6752.9	0.77%	5579	38.1
	20	6701.1	137	6874.3	2.52%	8167	59.8
	50	6701.1	135	6840.1	2.03%	14719	109.0
B + L	1	6701.1	70	6772.7	1.06%	467	7.1
	2	6701.1	56	6753.9	0.78%	632	14.8
	3	6701.1	38	6831.0	1.90%	546	15.7
	5	6701.2	34	6807.0	1.56%	752	20.8
	10	6701.0	24	6818.6	1.72%	737	32.7
	20	6700.9	23	6838.3	2.01%	952	39.1
	50	6701.1	21	6843.5	2.08%	1230	60.5
SB + I	1	6700.3	178	6808.1	1.58%	461	2.6
	2	6701.0	114	6825.9	1.82%	643	5.7
	3	6701.1	95	6800.6	1.46%	618	6.5
	5	6701.1	35	6768.4	0.99%	624	9.5
	10	6701.1	31	6763.0	0.91%	760	14.9
	20	6701.1	25	6803.9	1.51%	814	20.7
	50	6701.1	27	6860.6	2.32%	1239	32.4
SB + L	1	6701.0	61	6808.5	1.58%	401	6.6
	2	6701.0	40	6788.5	1.29%	457	11.6
	3	6701.0	33	6766.3	0.97%	496	14.9
	5	6701.1	29	6827.9	1.86%	621	21.8
	10	6701.0	22	6768.9	1.00%	611	28.1
	20	6701.1	20	6761.2	0.89%	767	37.7
	50	6701.1	20	6783.9	1.22%	1083	53.3
SB + I + L	1	6701.0	57	6800.5	1.46%	437	7.6
	2	6701.0	42	6763.5	0.92%	582	14.0
	3	6701.0	30	6817.1	1.70%	404	13.8
	5	6701.1	27	6783.4	1.21%	527	19.3
	10	6701.0	21	6835.1	1.96%	580	28.1
	20	6701.1	21	6796.8	1.41%	772	36.7
	50	6701.1	20	6813.3	1.65%	960	47.2

As we increase the number of sample paths evaluated in the forward step, the total computation time as well as the time used per iteration increase in general. The more scenarios are selected in the forward step, the more subproblems need to be solved, and it is often the case that more candidate solutions will be generated and evaluated in the backward step. A significant advantage of using only 1 sample path in the forward step was reported in [78]. Similar results can be observed in our experiments. Though for some instances (e.g., B +

I), a slightly bigger number (e.g., 3) of forward paths results in better performance of SDDiP algorithm. In general, the best choice of forward sample size remains small (1, 2, or 3). Moreover, in the experiments where Lagrangian cuts are used, the time used per iteration is usually longer. Since generating integer optimality cuts only requires solving the subproblem as an integer program, whereas one needs to solve a Lagrangian dual problem to get a Lagrangian cut, and the basic subgradient method usually takes more time.

To summarize, cut combinations (B + L), (SB + I), (SB + L), and (SB + I + L), appear to be good choices to be integrated into the SDDiP framework. In the case where the Lagrangian dual problem is difficult to solve, strengthened Benders' cuts and integer optimality cuts yield a better performance.

Scalability To further test the scalability of the algorithm, we generate several large-scale instances with planning horizons ranging from 5 to 9, and each period contains 30 to 50 realizations of the uncertain parameters, which are sampled independently from their distributions.The extensive scenario tree formulation (2.1) for these instances contains as many as 11 trillion binary variables, so it is impossible to expect any solver can solve such a problem as a single MIP. However, the SDDiP algorithm is able to estimate the optimal values of these instances with very high accuracy, as shown in Table 3.

Table 3: Performance of SDDiP algorithm on some large instances

T	# branch	cuts	best LB ($MM)	# iter	stat. UB ($MM)	gap	time (hr.)	time/iter (sec.)
5	50	B + I	2246.4	92	2260.7	0.63%	0.96	37.6
		SB + I	2246.4	34	2278.2	1.39%	0.09	9.4
		B + L	2246.4	34	2279.6	1.45%	0.19	20.3
		SB + L	2246.4	21	2276.4	1.32%	0.14	23.4
		SB + I + L	2246.4	25	2279.4	1.45%	0.11	15.4
6	50	B + I	2818.8	237	2840.6	0.77%	2.24	34.0
		SB + I	2818.9	74	2855.8	1.29%	0.60	29.0
		B + L	2818.9	63	2848.5	1.04%	0.96	54.7
		SB + L	2818.9	56	2849.2	1.06%	0.70	45.2
		SB + I + L	2818.9	50	2820.7	0.06%	1.03	73.9
7	50	B + I	3564.5	239	3614.8	1.39%	8.10	122.0
		SB + I	3564.4	111	3588.9	0.68%	1.08	34.9
		B + L	3564.5	100	3569.1	0.13%	2.48	89.2
		SB + L	3564.5	66	3576.9	0.35%	2.37	129.0
		SB + I + L	3564.5	69	3577.6	0.37%	1.95	101.6
8	30	B + I	4159.4	340	4254.2	2.23%	7.78	82.4
		SB + I	4159.4	152	4207.5	1.14%	1.53	36.3
		B + L	4159.6	147	4227.7	1.61%	4.00	97.9
		SB + L	4159.6	87	4218.9	1.41%	2.55	105.4
		SB + I + L	4159.6	103	4278.0	2.77%	2.72	94.9
9	30	B + I	5058.0	520	5081.5	0.46%	19.85	137.4
		SB + I	5058.6	230	5102.0	0.85%	2.57	40.2
		B + L	5058.7	218	5108.6	0.98%	8.01	132.3
		SB + L	5058.7	120	5145.3	1.68%	2.96	88.8
		SB + I + L	5058.9	119	5079.4	0.40%	4.39	132.8

In Table 3, Column 1 indicates the planning horizon of the corresponding instance, Column 2 shows the number of branches of each node in the scenario tree, and Column 3 indicates the cut combinations used in the backward step. In these instances, we do not enforce computation time limit, the algorithm stops when the lower bounds become stable. In all experiments, we achieve good estimates on the optimal value (small gaps between upper and lower bounds) within a reasonable computation time. Notice that the reduction in the number of iterations and computation time from cut combination (B + I) to (SB + I) or

(B + L) is significant. Moreover, the time per iteration is also significantly reduced even though SB and L require solving additional integer subproblems. This is perhaps because the later iterations, where more cuts are accumulated, take longer time, and using SB and L reduces the iteration count. The difficulty and time requirement for solving Lagrangian dual problems can be observed by comparing cut combination (SB + I + L) with (SB + I). Although the number of iterations decreases after adding Lagrangian cuts (which implies that these cuts provide better approximation than integer optimality cuts), both the total computation time and time used per iteration increase considerably. We finally point out that in all our experiments, the combination of strengthened Benders' cuts and integer optimality cuts (SB + I) outperforms other combinations in terms of total computation time.

These computational results demonstrate that the SDDiP algorithm with the proposed cuts successfully estimates the optimal value of large-scale generation capacity expansion problems with high accuracy and reasonable computation time.

6.2 Multi-period Portfolio Optimization

In this section, we test SDDiP algorithm on a multi-period portfolio optimization problem [see e.g., 25], where the uncertain parameters are the returns of different assets in each period. In this problem, the objective is to maximize the expected return over a fixed length of time periods, by adjusting the holding position of each type of asset. Each completed transaction will incur a certain amount of fee, referred as transaction cost, which is assumed to be a proportional cost to the total value of assets involved in the corresponding transaction. At any time period, the total number of assets possessed is restricted to be less than some prescribed threshold.

In particular, we consider n types of stocks and a risk-free asset (the $(n+1)$-th asset) over a T-period investment horizon. Let x_t be a vector denoting the values of assets at period t, and z_t be a binary vector, representing whether the account holder owns each asset at period t. The account holder decides how much of each stock to buy (b_t) or sell (s_t) at period t, with return information $r_0, \ldots, r_{t-1}$ which have been realized. We assume that the initial risk-free asset value is $\bar{x}_0$ and all others are 0. A deterministic model is as follows:

$$
\begin{aligned}
\max \quad & r_T^\top x_T \\
\text{s.t.} \quad & \forall t = 1, \ldots, T, \\
& x_{ti} = r_{t-1,i} x_{t-1,i} + b_{t,i} - s_{t,i} \;\; \forall i = 1, \ldots, n, && \text{(transaction flow balance)} \\
& x_{t,n+1} = r^f x_{t-1,n+1} - (\mathbf{1} + \alpha_b)^\top b_t + (\mathbf{1} - \alpha_s)^\top s_t, && \text{(self-financing)} \\
& x_t \le M z_t, \; s_{ti} \le r_{t-1,i} x_{t-1,i}, \; \forall i = 1, \ldots, n, && \text{(variable relationships)} \\
& \mathbf{1}^\top z_t \le K, && \text{(number of assets possessed)} \\
& x_0 = [0, \ldots, 0, \bar{x}_0]^\top, \\
& z_t \in \{0,1\}^n, 0 \le b_t, s_t \le u, 0 \le x_t \le v,
\end{aligned}
$$

where α_b and α_s are the transaction cost coefficients for buy and sell, respectively, and u, v are implied bounds on variables. For the stochastic model, the uncertainty is in the return vector r.

Scenario Generation We test the problem on all the stocks from the S&P 100 index. The optimization problem has an investment horizon of 5 to 12 periods, each of which is a two-week (10 business days) span. The scenarios of returns for each stock are generated using historical returns data without assuming specific distributions. In particular, we collect 500 bi-weekly returns over the past 2 years for each stock, and regard these 500 overlapping returns as the universe of all possible return realizations for each investment period. Then we sample (with replacement) a subset of realizations at each period independently to form a recombining scenario tree. To preserve the correlation between different stocks, the sampled scenario contains a return vector in which all components correspond to the same time span. In the scenario tree, the number of branches ranges from 10 to 20.

Note that in this problem, x_t are continuous state variables. We will resort to the binary approximation discussed in Section 2. We assume that at the beginning the account holder has 100 units of cash and none of

Table 4: Performance of SDDiP algorithm on portfolio optimization

T	# branch	# scen	# FW	Best UB	Stat. LB	gap	time (sec)
4	10	1000	1	108.1	105.7	0.66%	185
			2	108.1	106.4	1.33%	210
			5	108.1	106.3	1.41%	313
			10	108.1	106.1	1.00%	456
	15	3375	1	106.9	105.1	1.10%	309
			2	106.9	104.4	1.42%	356
			5	106.9	104.6	1.07%	518
			10	106.9	104.3	0.36%	884
	20	8000	1	108.1	106.2	1.05%	418
			2	108.1	106.1	1.63%	423
			5	108.1	105.0	1.25%	630
			10	108.1	106.1	1.49%	1027
5	10	10000	1	116.1	112.9	1.49%	343
			2	116.1	112.0	1.79%	414
			5	116.1	112.8	1.30%	580
	15	50625	1	109.6	106.9	1.65%	567
			2	109.6	106.6	0.98%	686
			5	109.6	106.3	2.07%	933
	20	160000	1	109.0	106.9	1.45%	425
			2	109.0	106.1	1.49%	715
			5	109.0	106.3	1.54%	1156
6	20	3.2×10^6	1	112.2	109.1	1.14%	704
			2	112.2	109.8	1.58%	1091
			5	112.2	108.2	2.08%	1573
7	20	6.4×10^7	1	116.5	112.8	1.71%	938
			2	116.5	112.8	1.24%	1201
			5	116.5	112.8	1.64%	2008
8	15	1.7×10^8	1	120.57	119.29	1.08%	1182
10	10	10^9	1	125.21	122.43	2.27%	1032
12	10	10^{11}	1	129.79	126.83	2.33%	1299

the stocks. The continuous state variables are approximated using the binary expansion with approximation accuracy $\varepsilon = 10^{-2}$. Each stage subproblem contains approximately 1500 binary state variables. The local variables are z_t, b_t, and s_t, each has a dimension of 100.

Algorithm Performance Table 4 summarizes the performance of SDDiP algorithm on the test instances. Since this is a maximization problem, the negation of the lower bound reported by SDDiP algorithm is a valid upper bound on the true optimal value (Column 5). The algorithm also produces a statistical lower bound on the expected return (Column 6), obtained by evaluating 500 sample paths independently after the upper bounds become stable. Column 1 shows the time horizon of the test instances; Columns 2 and 3 contain information of the scenario tree, i.e., number of branches of each node and total number of scenarios; Column 4 indicates how many forward samples paths are used in the forward step; Columns 7 and 8 report the gaps between the lower and upper bounds on the optimal values, and the total computation time, respectively.

The stopping criterion remains the same, i.e., the algorithm stops when the deterministic upper bounds become stable. Among all test instances, the algorithm reaches the stopping criterion within 10 iterations, and gaps between the upper bound and the statistical lower bound are all small. We solve the extensive scenario tree formulation for the first two instances $T = 4$, #branch $= 10$ and 15 as two examples to demonstrate the accuracy of attained upper bounds. The first instance has an optimal value of 108.02 and the second is 106.8. The gap between the lower and upper bounds mostly come from the evaluation of lower bounds, and can be made smaller by evaluating more forward paths. Similar to the generation capacity expansion example, we observe that it is more efficient to use a small number of sample paths in the forward iteration. Note that we generate a different scenario tree for each instance (T, #branch), thus the optimal values are not necessary monotone.

6.3 Airline Revenue Management

In the airline industry, revenue management usually refers to dynamic pricing and controlling seat sales based on the passenger demand forecast in a flight network. In this section, we focus on the latter approach. The objective is to maximize the revenue generated from ticket sales. We consider a multistage stochastic model which is similar to the one in [54]. A deterministic formulation of such a problem is given as follows.

$$\begin{aligned}
\max \quad & \sum_{t=1}^{T} \left[(f_t^b)^\top b_t - (f_t^c)^\top c_t \right] \\
\text{s.t.} \quad & \forall t = 1, \dots, T, \\
& B_t = B_{t-1} + b_t,\ C_t = C_{t-1} + c_t \\
& C_t = \lfloor \Gamma_t B_t + 0.5 \rfloor \\
& A(B_t - C_t) \le R,\ b_t \le d_t \\
& B_0 = \bar{B}_0,\ C_0 = \bar{C}_0 \\
& B_t, C_t, b_t, c_t \in \mathbb{Z}_+^m.
\end{aligned}$$

In the above formulation, T is the number of booking intervals. The numbers of fulfilled bookings (resp. cancellations) of period t and cumulative fulfilled bookings (resp. cancellations) up to period t are denoted by b_t (resp. c_t) and B_t (resp. C_t). Each of these quantities is an m-dimensional vector, whose components correspond to particular origin-destination itineraries and fare classes. f_t^b and f_t^c are the booking price and refund for cancellation at period t, respectively. The matrix Γ_t is a diagonal matrix, whose elements are the cancellation rate of each type of tickets. Passenger demand is denoted by d_t, which is subject to uncertainty. The seat capacity on each leg is denoted by R, and A is a 0-1 matrix that indicates whether a booking request for a particular itinerary and fare class fills up one unit of capacity of each leg.

Scenario Generation The underlying flight network contains a single hub and three destinations. There are in total 6 legs and 12 itineraries. Ticket prices and refund are fixed over booking intervals. Cancellation rates for different fare classes are also given as constants. All data can be found in [54]. As proposed in the literature [see e.g., 26; 22], the booking process is modeled by a non-homogeneous Poisson process. The total number of cumulative booking request G over the entire booking horizon for a particular itinerary and fare class is assumed to follow a Gamma distribution $G \sim \Gamma(k, \theta)$, and the corresponding arrival pattern β follows a Beta distribution $\beta \sim \text{Beta}(a, b)$. The arrival pattern determines an allocation of total booking requests among booking intervals. The cumulative booking requests up to time $t \in [0, T]$ can be represented by $D(t) = G \cdot F_\beta(t, a, b)$, where $F_\beta(t, a, b)$ is the cumulative density function of the Beta distribution. We generate the scenario tree as follows. First, we generate N_0 realizations for the cumulative booking request for each itinerary and class fare combination, and allocate them according to the corresponding arrival patterns into each booking interval. Then, for each booking interval, N_b samples are drawn independently out of the N_0 realizations, where N_b is the number of branches of each node in the scenario tree. In this way, we obtain a recombining scenario tree which preserves stage-wise independece. It has T stages, each of which contains N_b nodes, hence there are N_b^{T-1} scenarios in total.

In this problem, the state variables are B_t and C_t, and local variables are b_t and c_t. After binary expansion, the stage problem contains about 3000 binary state variables, and the local variables are general integers with dimension 144.

Algorithm Performance We divide the booking horizon of 182 days into different numbers of booking intervals (stages), from 6 to 14 (not necessarily evenly divided), and generate scenario trees separately for each of them. The scenario tree information is contained in the first three columns of Table 5. We test SDDiP algorithm on these 5 instances. During the experiment, we notice that the stage subproblem is more difficult to solve than in the previous two examples, hence we relax the relative MIP tolerance from the default (10^{-4}) to 0.05. In addition, we enforce limits on both the number of total iterations (120) and computation time (5 hours).

Table 5: Performance of SDDiP algorithm on network revenue management

T	# branch	# scen	# iter	Best UB	Stat. LB	gap	time (sec)
6	10	10^5	120	214357	204071	5.04%	10983
8	10	10^7	120	214855	201099	6.84%	12095
10	10	10^9	120	215039	199896	7.58%	14674
12	10	10^{11}	120	210110	196237	7.07%	15413
14	10	10^{13}	120	210012	196280	7.00%	15241

Table 5 summarizes the results for these 5 instances. All of them terminate because of reaching the limit on number of iterations. We observe relatively larger but acceptable gaps between the lower and upper bounds on the optimal values. These relatively larger gap could be a consequence of early termination due to the difficulty of solving the stage problems, or possibly because the 5% relative MIP error accumulated over the stages. We would also like to note that, due to the very large scale of the underlying multistage stochastic programs, the extensive form problems can not be solved by existing solvers. Therefore, the SDDiP algorithm with proposed cuts provides a viable and systematic way to tackle these extremely challenging problems in network revenue management.

7 Concluding Remarks

We consider a large class of multistage stochastic integer programs in which the variables that carry information from one stage to the next are purely binary. By exploiting the binary nature of the state variables, we propose a stochastic nested decomposition algorithm and a stochastic dual dynamic integer programming algorithm. We remark that the binary feature of the state variables and making a local copy of state variables are the key elements to the success of the approach. It allows us to construct supporting hyperplanes to the expected cost-to-go functions, which is crucial for the correctness of the method. Extensive computational experiments on three classes of real-world problems, namely electric generation expansion, financial portfolio management, and network revenue management, show that the proposed methodology may lead to significant improvement on solving large-scale, multistage stochastic optimization problems in real-world applications.

There are several interesting directions worth investigating for future research. Improvements to the integer optimality cut for two-stage stochastic integer programs are recently proposed in [7], and this may be considered for extension to the multistage setting. In addition, it would also be interesting to see the computation time improvement if the Lagrangian dual problem is solved by a more advanced methods, such as bundle method [41] or column generation [9]. Since we have observed that the stage problem is sometimes not very easy to solve, to further improve performance, one needs to explore the problem substructure and tailor the algorithm according to specific problems. Effective cut management strategies could be explored to keep the problem sizes small, especially in later iterations. Finally, extension of the proposed approach to the risk averse setting would be valuable. Most previous work in risk averse multistage stochastic programming is restricted to the linear or convex settings [75; 77; 66; 78; 17], it is intriguing to study how the nonlinearity of risk in the presence of integer variables affect the problem structure.

Appendix

Proof of Theorem 3. Consider an MSIP with $d := d_1 + d_2$ mixed-integer state variables per node:

$$\begin{array}{lll} \min\limits_{x_n, y_n} & \sum\limits_{n\in\mathcal{T}} p_n f_n(x_n, y_n) & \\ \text{s.t.} & (x_{a(n)}, x_n, y_n) \in X_n & \forall\, n \in \mathcal{T} \\ & x_n \in \mathbb{Z}_+^{d_1} \times \mathbb{R}_+^{d_2} & \forall\, n \in \mathcal{T}. \end{array} \tag{7.1}$$

Since the state variables are bounded by (A1), we can assume that $x_n \in [0, U]^d$ for some positive integer U for all $n \in \mathcal{T}$.

We approximate (7.1) as follows. For an integer state variable $x \in \{0, \ldots, U\}$, we substitute by its binary expansion: $x = \sum_{i=1}^{\kappa} 2^{i-1}\lambda_i$ where $\lambda_i \in \{0,1\}$ and $\kappa = \lfloor \log_2 U \rfloor + 1$. For a continuous state variable $x \in [0, U]$, we approximate it by binary approximation to a precision of $\epsilon \in (0,1)$, i.e. $x = \sum_{i=1}^{\kappa} 2^{i-1}\epsilon\lambda_i$ where $\lambda_i \in \{0,1\}$ and $\kappa = \lfloor \log_2(U/\epsilon) \rfloor + 1$ [see e.g., 34]. Note that $|x - \sum_{i=1}^{\kappa} 2^{i-1}\epsilon\lambda_i| \le \epsilon$. The total number k of binary variables introduced to approximate the d state variables thus satisfies $k \le d(\lfloor \log_2(U/\epsilon) \rfloor + 1)$. We then have the following approximating MSIP with binary variables $\lambda_n \in \{0,1\}^k$

$$\begin{array}{lll} \min\limits_{\lambda_n, y_n} & \sum\limits_{n\in\mathcal{T}} p_n f_n(A\lambda_n, y_n) & \\ \text{s.t.} & (A\lambda_{a(n)}, A\lambda_n, y_n) \in X_n & \forall\, n \in \mathcal{T} \\ & \lambda_n \in \{0,1\}^k & \forall\, n \in \mathcal{T}, \end{array} \tag{7.2}$$

where the $d \times k$ matrix A encodes the coefficients of the binary expansion.

Recall that the local variables are mixed integer, i.e. $y_n = (u_n, v_n)$ with $u_n \in \mathbb{Z}_+^{\ell_1}$ and $v_n \in \mathbb{R}_+^{\ell_2}$. Given $x := \{x_n \in \mathbb{Z}^{d_1} \times \mathbb{R}^{d_2}\}_{n\in\mathcal{T}}$, let

$$\begin{aligned} \phi(x) &:= \min_{u,v} \left\{ \sum_{n\in\mathcal{T}} f_n(x_n, (u_n, v_n)) : (x_{a(n)}, x_n, (u_n, v_n)) \in X_n,\ \forall n \in \mathcal{T} \right\} \\ &= \sum_{n\in\mathcal{T}} \min_{u_n, v_n} \left\{ f_n(x_n, (u_n, v_n)) : (x_{a(n)}, x_n, (u_n, v_n)) \in X_n \right\} \\ &= \sum_{n\in\mathcal{T}} \min_{u_n \in \mathcal{U}_n} \left\{ \psi_n(x_{a(n)}, x_n, u_n) \right\}, \end{aligned}$$

where

$$\psi_n(x_{a(n)}, x_n, u_n) = \min_{v_n \in \mathbb{R}_+^{\ell_2}} \left\{ f_n(x_n, (u_n, v_n)) : (x_{a(n)}, x_n, (u_n, v_n)) \in X_n \right\},$$

and $\mathcal{U}_n$ is the finite set of integer values the local variable u_n can take. By the compactness assumption (A1) and the complete continuous recourse assumption (A2), the function ψ_n is the value function of a linear program that is feasible and bounded for all values of $(x_{a(n)}, x_n, u_n)$. By Hoffman's lemma [42], there exists a constant $C_n(u_n)$ which is dependent on the data defining (f_n, X_n) and u_n, such that $\psi_n(x_{a(n)}, x_n, u_n)$ is Lipschitz continuous with respect to $(x_{a(n)}, x_n)$ with this constant. It follows that $\phi(x)$ is Lipschitz continuous with respect to x with constant $C = \sum_{n\in T} \max_{u_n \in U_n} C_n(u_n)$, i.e.,

$$|\phi(x) - \phi(x')| \le C\|x - x'\| \ \ \forall\, x, x'.$$

Let $(\tilde{\lambda}, \tilde{y})$ be an optimal solution to problem (7.2) and v_2 be its optimal value. Define $\tilde{x}_n = A\tilde{\lambda}_n$ for all $n \in \mathcal{T}$, then $(\tilde{x}, \tilde{y})$ is a feasible solution to (7.1) and has the objective value of v_2. From the definition of ϕ we have that $v_2 = \phi(\tilde{x})$. Now let $(\hat{x}, \hat{y})$ be an optimal solution of (7.1) and v_1 be its optimal value. Note that $v_1 = \phi(\hat{x})$. Let us construct a solution $(\hat{\lambda}, \hat{y}')$ such that

$$\|\hat{x} - A\hat{\lambda}\| \le \epsilon, \ \text{ and } \ \hat{y}'_n = \text{argmin}_{y_n} \left\{ f(A\hat{\lambda}_{a(n)}, A\hat{\lambda}_n, y_n) : (A\hat{\lambda}_{a(n)}, A\hat{\lambda}_n, y_n) \in X_n \right\}.$$

Then $(\hat{\lambda}, \hat{y}')$ is clearly a feasible solution to (7.2) and has the objective value $\phi(A\hat{\lambda})$. Thus we have the following inequalities

$$\phi(\hat{x}) \leq \phi(\tilde{x}) \leq \phi(A\hat{\lambda}).$$

Thus

$$0 \leq \phi(\tilde{x}) - \phi(\hat{x}) \leq |\phi(A\hat{\lambda}) - \phi(\hat{x})| \leq C\|A\hat{\lambda} - \hat{x}\| \leq C\epsilon.$$

By choosing $\epsilon = \varepsilon/C$ and $M = UC$ we have that $(\tilde{x}, \tilde{y})$ is a ε-optimal solution of (7.1) and $k \leq d(\lfloor \log_2(M/\varepsilon) \rfloor + 1)$ as desired. □

References

[1] H. Abgottspon, K. Njalsson, M. Bucher, G. Andersson, et al. Risk-averse medium-term hydro optimization considering provision of spinning reserves. In *Probabilistic Methods Applied to Power Systems (PMAPS), 2014 International Conference on*, pages 1–6. IEEE, 2014.

[2] S. Ahmed. Two-stage stochastic integer programming: A brief introduction. In *Wiley Encyclopedia of Operations Research and Management Science,* Cochran et al. (Eds), 2010.

[3] S. Ahmed and N. V. Sahinidis. An approximation scheme for stochastic integer programs arising in capacity expansion. *Operations Research*, 51(3):461–471, 2003.

[4] S. Ahmed, A. J. King, and G. Parija. A multi-stage stochastic integer programming approach for capacity expansion under uncertainty. *Journal of Global Optimization*, 26(1):3–24, 2003.

[5] T. Akbari, A. Rahimikian, and A. Kazemi. A multi-stage stochastic transmission expansion planning method. *Energy Conversion and Management*, 52(8):2844–2853, 2011.

[6] A. Alonso-Ayuso, L. F. Escudero, and M. T. Ortuno. Bfc, a branch-and-fix coordination algorithmic framework for solving some types of stochastic pure and mixed 0–1 programs. *European Journal of Operational Research*, 151(3): 503–519, 2003.

[7] G. Angulo, S. Ahmed, and S. S. Dey. Improving the integer L-shaped method. *INFORMS Journal on Computing*, 28: 483–399, 2016.

[8] L. Baringo and A. J. Conejo. Risk-constrained multi-stage wind power investment. *Power Systems, IEEE Transactions on*, 28(1):401–411, 2013.

[9] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[10] R. Barth, H. Brand, P. Meibom, and C. Weber. A stochastic unit-commitment model for the evaluation of the impacts of integration of large amounts of intermittent wind power. In *Probabilistic Methods Applied to Power Systems, 2006. PMAPS 2006. International Conference on*, pages 1–8. IEEE, 2006.

[11] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4 (1):238–252, 1962.

[12] D. P. Bertsekas. *Nonlinear programming*. Athena scientific, 1999.

[13] D. Bienstock and G. Munoz. LP approximations to mixed-integer polynomial optimization problems, 2016. arXiv:1501.00288.

[14] J. R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research*, 33(5):989–1007, 1985.

[15] N. Boland, I. Dumitrescu, G. Froyland, and T. Kalinowski. Minimum cardinality non-anticipativity constraints sets for multistage stochastic programming. *Mathematical Programming*, 157(2):69–93, 2016.

[16] S. P. Bradley and D. B. Crane. A dynamic model for bond portfolio management. *Management Science*, 19(2):139–151, 1972.

[17] S. Bruno, S. Ahmed, A. Shapiro, and A. Street. Risk neutral and risk averse approaches to multistage renewable investment planning under uncertainty. *European Journal of Operational Research*, 250(3):979–989, 2016.

[18] D. R. Carino, T. Kent, D. H. Myers, C. Stacy, M. Sylvanus, A. L. Turner, K. Watanabe, and W. T. Ziemba. The russell-yasuda kasai model: An asset/liability model for a japanese insurance company using multistage stochastic programming. *Interfaces*, 24(1):29–49, 1994.

[19] C. C. CarøE and R. Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24 (1):37–45, 1999.

[20] S. Cerisola, Á. Baíllo, J. M. Fernández-López, A. Ramos, and R. Gollmer. Stochastic power generation unit commitment in electricity markets: A novel formulation and a comparison of solution methods. *Operations Research*, 57(1):32–46, 2009.

[21] S. Cerisola, J. M. Latorre, and A. Ramos. Stochastic dual dynamic programming applied to nonconvex hydrothermal models. *European Journal of Operational Research*, 218(3):687–697, 2012.

[22] L. Chen and T. Homem-de Mello. Re-solving stochastic programming models for airline revenue management. *Annals of Operations Research*, 177(1):91–114, 2010.

[23] Z.-L. Chen and W. B. Powell. Convergent cutting-plane and partial-sampling algorithm for multistage stochastic linear programs with recourse. *Journal of Optimization Theory and Applications*, 102(3):497–524, 1999.

[24] Z.-L. Chen, S. Li, and D. Tirupati. A scenario-based stochastic programming approach for technology and capacity planning. *Computers & Operations Research*, 29(7):781–806, 2002.

[25] G. B. Dantzig and G. Infanger. Multi-stage stochastic linear programs for portfolio optimization. *Annals of Operations Research*, 45(1):59–76, 1993.

[26] S. V. de Boer, R. Freling, and N. Piersma. Mathematical programming for network revenue management revisited. *European Journal of Operational Research*, 137(1):72–92, 2002.

[27] L. F. Escudero, P. V. Kamesam, A. J. King, and R. J. Wets. Production planning via scenario modelling. *Annals of Operations Research*, 43(6):309–335, 1993.

[28] L. F. Escudero, A. Garin, and A. Unzeuta. Cluster lagrangean decomposition in multistage stochastic optimization. *Computers & Operations Research*, 67:48–62, 2016.

[29] B. Flach, L. Barroso, and M. Pereira. Long-term optimal allocation of hydro generation for a price-maker company in a competitive market: latest developments and a stochastic dual dynamic programming approach. *IET generation, transmission & distribution*, 4(2):299–314, 2010.

[30] S.-E. Fleten and T. K. Kristoffersen. Short-term hydropower production planning by stochastic programming. *Computers & Operations Research*, 35(8):2656–2671, 2008.

[31] D. Gade, G. Hackebeil, S. Ryan, J.-P. Watson, R. Wets, and D. L. Woodruff. Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs. *Mathematical Programming*, 157(1):47–67, 2016.

[32] P. Girardeau, V. Leclere, and A. Philpott. On the convergence of decomposition methods for multistage stochastic convex programs. *Mathematics of Operations Research*, 40(1):130–145, 2014.

[33] A. Gjelsvik, M. M. Belsnes, and A. Haugstad. An algorithm for stochastic medium-term hydrothermal scheduling under spot price uncertainty. In *Proceedings of 13th Power Systems Computation Conference*, 1999.

[34] F. Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22(4):455–460, 1975.

[35] B. Golub, M. Holmer, R. McKendall, L. Pohlman, and S. A. Zenios. A stochastic programming model for money management. *European Journal of Operational Research*, 85(2):282–296, 1995.

[36] V. Gupta and I. E. Grossmann. Multistage stochastic programming approach for offshore oilfield infrastructure planning under production sharing agreements and endogenous uncertainties. *Journal of Petroleum Science and Engineering*, 124:180–197, 2014.

[37] A. Gupte, S. Ahmed, M. Cheon, and S. Dey. Solving mixed integer bilinear problems using MILP formulations. *SIAM Journal on Optimization*, 23:721–744, 2013, 2013.

[38] A. Gupte, S. Ahmed, M. Cheon, and S. Dey. Relaxations and discretizations for the pooling problem. *Journal of Global Optimization*, 67:631–669, 2017.

[39] H. Heitsch, W. Römisch, and C. Strugarek. Stability of multistage stochastic programs. *SIAM Journal on Optimization*, 17(2):511–525, 2006.

[40] A. Helseth, B. Mo, M. Fodstad, and M. N. Hjelmeland. Co-optimizing sales of energy and capacity in a hydropower scheduling model. In *PowerTech, 2015 IEEE Eindhoven*, pages 1–6. IEEE, 2015.

[41] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms I: Fundamentals*, volume 305. Springer Science & Business Media, 2013.

[42] A. J. Hoffman. On approximate solutions of systems of linear inequalities. *Journal of Research of the National Bureau of Standards*, 49(4):263–265, 1952.

[43] K. Høyland and S. W. Wallace. Generating scenario trees for multistage decision problems. *Management Science*, 47 (2):295–307, 2001.

[44] J. Jacobs, G. Freeman, J. Grygier, D. Morton, G. Schultz, K. Staschus, and J. Stedinger. Socrates: A system for scheduling hydroelectric generation under uncertainty. *Annals of Operations Research*, 59(1):99–133, 1995.

[45] S. Jin, S. M. Ryan, J.-P. Watson, and D. L. Woodruff. Modeling and solving a large-scale generation expansion planning problem under uncertainty. *Energy Systems*, 2(3-4):209–242, 2011.

[46] D. Kuhn. *Generalized bounds for convex multistage stochastic programs*, volume 548. Springer Science & Business Media, 2006.

[47] M. I. Kusy and W. T. Ziemba. A bank asset and liability management model. *Operations Research*, 34(3):356–376, 1986.

[48] G. Laporte and F. V. Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations research letters*, 13(3):133–142, 1993.

[49] Y. Li, G. Huang, S. Nie, and L. Liu. Inexact multistage stochastic integer programming for water resources management under uncertainty. *Journal of Environmental Management*, 88(1):93–107, 2008.

[50] N. Löhndorf, D. Wozabal, and S. Minner. Optimizing trading decisions for hydro storage systems using approximate dual dynamic programming. *Operations Research*, 61(4):810–823, 2013.

[51] Y. Lu, C. Zhao, J.-P. Watson, K. Pan, and Y. Guan. Two-stage and multi-stage stochastic unit commitment under wind generation uncertainty. In *Proceedings of the IEEE PES Annual Conference*, 2014.

[52] P. Meibom, R. Barth, B. Hasche, H. Brand, C. Weber, and M. O'Malley. Stochastic optimization model to study the operational impacts of high wind penetrations in ireland. *Power Systems, IEEE Transactions on*, 26(3):1367–1379, 2011.

[53] P. Mokrian and M. Stephen. A stochastic programming framework for the valuation of electricity storage. In *26th USAEE/IAEE North American Conference*, pages 24–27, 2006.

[54] A. Möller, W. Römisch, and K. Weber. Airline network revenue management by multistage stochastic programming. *Computational Management Science*, 5(355–377), 2008.

[55] J. M. Mulvey and H. Vladimirou. Stochastic network programming for financial planning problems. *Management Science*, 38(11):1642–1664, 1992.

[56] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. John Wiley & Sons, 2014.

[57] N. Newham and A. Wood. Transmission investment planning using sddp. In *Power Engineering Conference, 2007. AUPEC 2007. Australasian Universities*, pages 1–5. IEEE, 2007.

[58] M. P. Nowak and W. Römisch. Stochastic lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty. *Annals of Operations Research*, 100(1-4):251–272, 2000.

[59] J. Owen and S. Mehrotra. On the value of binary expansions for general mixed- integer linear programs. *Operations Research*, 50:810–819, 2002.

[60] V. S. Pappala, I. Erlich, K. Rohrig, and J. Dobschinski. A stochastic model for the optimal operation of a wind-thermal power system. *Power Systems, IEEE Transactions on*, 24(2):940–950, 2009.

[61] T. Pennanen. Epi-convergent discretizations of multistage stochastic programs via integration quadratures. *Mathematical Programming*, 116(1-2):461–479, 2009.

[62] M. V. Pereira and L. M. Pinto. Stochastic optimization of a multireservoir hydroelectric system: a decomposition approach. *Water Resources Research*, 21:779–792, 1985.

[63] M. V. Pereira and L. M. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1-3):359–375, 1991.

[64] G. C. Pflug. Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical programming*, 89(2):251–271, 2001.

[65] A. Philpott, F. Wahid, and B. Frédéric. MIDAS: A mixed integer dynamic approximation scheme. *Optimization-online*, 2016.

[66] A. B. Philpott and V. L. de Matos. Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion. *European Journal of Operational Research*, 218(2):470–483, 2012.

[67] A. B. Philpott and Z. Guan. On the convergence of stochastic dual dynamic programming and related methods. *Operations Research Letters*, 36(4):450–455, 2008.

[68] S. Rebennack. Combining sampling-based and scenario-based nested benders decomposition methods: application to stochastic dual dynamic programming. *Mathematical Programming*, pages 1–47, 2013.

[69] R. T. Rockafellar and R. Wets. Scenario and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16:119–147, 1991.

[70] W. Römisch and R. Schultz. Multistage stochastic integer programs: An introduction. In *Online optimization of large scale systems*, pages 581–600. Springer, 2001.

[71] A. Ruszczynski and A. Shapiro. *Stochastic programming*, volume 10. Elsevier Amsterdam, 2003.

[72] B. Sandikci and O. Y. Ozaltin. A scalable bounding method for multistage stochastic integer programs. *Working paper 14-21, Booth School of Business, University of Chicago*, 2014.

[73] S. Sen, L. Yu, and T. Genc. A stochastic programming approach to power portfolio optimization. *Operations Research*, 54(1):55–72, 2006.

[74] A. Shapiro. Inference of statistical bounds for multistage stochastic programming problems. *Mathematical Methods of Operations Research*, 58(1):57–68, 2003.

[75] A. Shapiro. On a time consistency concept in risk averse multistage stochastic programming. *Operations Research Letters*, 37(3):143–147, 2009.

[76] A. Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209 (1):63–72, 2011.

[77] A. Shapiro. Minimax and risk averse multistage stochastic programming. *European Journal of Operational Research*, 219(3):719–726, 2012.

[78] A. Shapiro, W. Tekaya, J. P. da Costa, and M. P. Soares. Risk neutral and risk averse stochastic dual dynamic programming method. *European journal of operational research*, 224(2):375–391, 2013.

[79] K. J. Singh, A. B. Philpott, and R. K. Wood. Dantzig-wolfe decomposition for solving multistage stochastic capacity-planning problems. *Operations Research*, 57(5):1271–1286, 2009.

[80] S. Takriti and J. R. Birge. Lagrangian solution techniques and bounds for loosely coupled mixed-integer stochastic programs. *Operations Research*, 48(1):91–98, 2000.

[81] S. Takriti, J. R. Birge, and E. Long. A stochastic model for the unit commitment problem. *Power Systems, IEEE Transactions on*, 11(3):1497–1508, 1996.

[82] S. Takriti, B. Krasenbrink, and L. S.-Y. Wu. Incorporating fuel constraints and electricity spot prices into the stochastic unit commitment problem. *Operations Research*, 48(2):268–280, 2000.

[83] F. Thomé, M. Pereira, S. Granville, and M. Fampa. Non-convexities representation on hydrothermal operation planning using sddp. *URL: www. psr-inc. com, submitted*, 2013.

[84] D. W. Watkins, D. C. McKinney, L. S. Lasdon, S. S. Nielsen, and Q. W. Martin. A scenario-based stochastic programming model for water supplies from the highland lakes. *International Transactions in Operational Research*, 7 (3):211–230, 2000.

[85] G. L. Zenarosa, O. A. Prokopyev, and A. J. Schaefer. Scenario-tree decomposition: Bounds for multistage stochastic mixed-integer programs. *Working paper, Department of Industrial Engineering, University of Pittsburgh*, 2014.