Proceedings of the 2019 IISE Annual Conference
H.E. Romeijn, A. Schaefer, R. Thomas, eds.

Adaptive Partition-enabled Preprocessing for Multistage Stochastic Linear Programs

Abstract ID: 610948

Abstract

We extend the adaptive partition-based approach for solving two-stage stochastic programs with fixed recourse to its multistage counterpart. The proposed algorithm integrates the adaptive partition-based strategy with a popular approach for solving multistage stochastic programs, *Stochastic Dual Dynamic Programming*, via a preprocessing step as a mean to enhance its convergence. Our numerical experiments on a hydro-thermal power generation planning problem show the effectiveness of the proposed approach.

Keywords
Partition-based approach, Stochastic dual dynamic programming, Multistage stochastic linear programs.

1. Introduction
In this paper, we are concerned with the computational efficiency of solving Multistage Stochastic Linear Programming Problems (MSLPs). As a classical optimization modeling tool, MSLPs has been widely applied in real-life applications such as energy [1], transportation [2] and finance [3], where decision makers have to cope with uncertainty regarding information about the future in a sequential fashion. MSLPs provide the framework for finding an optimal policy over a finite planning horizon — typically discretized into T intervals that allows for meaningful optimization problem to arise. To put this into context, in this work, we focus on a multistage hydro-thermal power generation planning problem over a certain period of time. The objective is to minimize the (expected) total cost imposed by the power generation expenses and the penalty for the insufficiency in satisfying the power demand, under the uncertainty on the amount of rainfall in the future. We refer readers to [4] for a detailed description of the problem. A common approach to tackle these problems is to approximate the underlying stochastic process by using a scenario tree which typically yields huge-scale mathematical programming models due to the infinite number of sample paths that can take place in the future. Despite the fact that these problems can be handled using specialized algorithms which employ decomposition techniques that often take advantage of the fact that the probability distributions governing the uncertainty are known, the computational burden for solving these problems increases significantly as we desire more accuracy in the approximation due to the *curse of dimensionality* [5].
In the literature, there is a good deal of work being done on mitigating this burden by using cuts-sharing strategies when the stochastic process is assumed to have a stage-wise independence structure. A milestone in doing so is the *stochastic dual dynamic programming* algorithm (SDDP), proposed in [6] which was proved to have an almost-sure convergence in [7]. Several algorithmic techniques were proposed to accelerate the convergence in SDDP such as the ones mentioned in [8, 9], which deploy regularization scheme that stabilizes iterates during the solution process.

A partition-based formulation, which was first introduced in [10] and later improved in [11], is a relaxation of the original stochastic program obtained by aggregating variables and constraints according to a partition over the set of scenarios. The partition is iteratively refined by exploiting the information of the dual multipliers associated with each scenario with respect to the relaxation solution until it is sufficient to induce an optimal solution to the original problem. We refer readers to section 3 for a general overview and [10, 11] for a detailed discussion on the topic.

The motivation behind considering the partition-based approach in the multistage setting is its promising computational performance in the two-stage context. However, as it turns out, constructing such partition for the multistage case might be intractable – a partition which deemed to be sufficient ought to accommodate all sample paths in the scenario tree, which leads back to the core difficulty in this problem. In this paper, we intend to cope with this difficulty by utilizing the adaptive partition-based approach as a catalyst in a pre-processing stage which is followed by the SDDP algorithm. Our approach generates a sequence of provisional partitions which are exploited to generate validly but coarse cuts to enhance the progress of the standard SDDP algorithm.

The rest of this paper is organized as follows. Section 2 introduces a general MSLP formulation and provides an overview of the SDDP algorithm. Section 3 reviews the adaptive partition-based approach. In Section 4 we present our proposed algorithm which integrates the adaptive partition-based approach with the SDDP algorithm as a preprocessing step. In Section 5, we provide some preliminary computational results for the proposed algorithm and compare its performance to the standard SDDP. Finally, in Section 6 we conclude with some final remarks.

2. Preliminaries on multistage stochastic linear programs and decomposition schemes

In this section we will provide an overview for one of the most popular decomposition approaches for MSLPs, *Stochastic Dual Dynamic Programming* (SDDP) [6], which we aim to improve. The main feature in SDDP is that it underestimates the cost-to-go functions (functions that quantify the expected impact of the current decision in the future) using piecewise linear approximations, defined by cutting planes which are generated in the so-called *backward step*. This can be seen as a variant of Kelley's cutting plane method [12] in non-linear non-smooth optimization. Consistently with [6], we make the conventional assumption that the random data process is stage-wise independent, i.e., the random vector which captures the uncertainty, denoted by $\xi_t \in \Xi_t$, is independent of the history of the data process $\xi_{[t-1]} := (\xi_1, ..., \xi_{t-1})$ up to time $t-1$. To put this into perspective, we consider the following MSLP:

$$\min_{\substack{A_1x_1=b_1\\ x_1\in\mathbb{R}^{n_1}_+}} c_1^\top x_1 + \mathbb{E}\left[\min_{\substack{B_2x_1+A_2x_2=b_2\\ x_2\in\mathbb{R}^{n_2}_+}} c_2^\top x_2 + \mathbb{E}\left[\cdots + \mathbb{E}\left[\min_{\substack{B_Tx_{T-1}+A_Tx_T=b_T\\ x_T\in\mathbb{R}^{n_T}_+}} c_T^\top x_T\right]\right]\right], \tag{1}$$

where some (if not all) data $\xi = (c_t, B_t, A_t, b_t)$, $\forall\, t = 2, \ldots, T$ can be subject to uncertainty. For the sake of numerical tractability, we assume that the number of realizations of the stochastic process is finite, i.e., $|\Xi_t| < \infty$, $\forall\, t = 2, \ldots, T$, and therefore can be represented using a scenario tree. This is, for example, the case in which (1) is a sample average approximation of an underlying MSLP with continuously distributed random variables. Under these assumptions, the dynamic programming equation for problem (1) at time $t = 2, \ldots, T$ takes the form:

$$Q_t(x_{t-1}, \xi_t) := \begin{cases} \min\limits_{x_t\in\mathbb{R}^{n_t}_+} & c_t^\top x_t + \mathfrak{Q}_{t+1}(x_t) \\ \text{s.t.} & A_t x_t = b_t - B_t x_{t-1}\,, \end{cases} \tag{2}$$

where $\mathfrak{Q}_{t+1}(x_t) := \mathbb{E}[Q_{t+1}(x_t, \xi_{t+1})]$, $\forall\, t = 1, \ldots T-1$ represents the expected cost-to-go functions, and $\mathfrak{Q}_{T+1}(x_T) := 0$. Additionally, as in [13], we assume *relatively complete recourse* for simplicity of presentation.

As previously noted, SDDP aims to approximate the cost-to-go function which induces a policy for decision making in each stage. To this end, the SDDP algorithm alternates between the two main steps. A forward simulation which evaluates the current policy induced by the current approximate cost-to-go functions, and a backward recursion to update the inaccuracy in the current approximation of the cost-to-go functions (if any). We summarize these two steps below and refer readers to [6] for a more detailed discussion.

Forward step. Given a sample path $\xi = (\xi_1, \ldots, \xi_T) \in \Xi_1 \times \ldots \times \Xi_T$, trial decisions $\bar{x}_t = \bar{x}_t(\xi_{[t]})$ are computed recursively going forward with $\bar{x}_t$ being an optimal solution of (2), $\forall\, t = 1, \ldots, T$ and using the current approximation $\check{\mathfrak{Q}}_t(\cdot)$ of the true expected cost-to-go function $\mathfrak{Q}_t(\cdot)$, which is unknown unless $t = T$. The approximate cost-to-go functions $\check{\mathfrak{Q}}_t(\cdot)$ induce an implementable policy for problem (1) via a set of trial decisions $\bar{x}_t(\xi_{[t]})$ for each sample path, to which case the value $\bar{z} = \mathbb{E}\left[\sum_{t=1}^T c_t^\top \bar{x}_t(\xi_{[t]})\right]$ gives an upper bound for the optimal value of (1) as long as all scenarios $\xi \in \Xi$ are considered. Nonetheless, the forward step of SDDP consists of taking a sample $\mathcal{J}$ with $|\mathcal{J}| \ll N$ scenarios $\{\xi^j\}_{j\in\mathcal{J}}$ of the stochastic process and computing $\bar{x}_t(\xi^j_{[t]})$ and the respective values $z(\xi^j) = \sum_{t=1}^T c_t^\top \bar{x}_t(\xi^j_{[t]})$, $\forall\, j \in \mathcal{J}$. Statistical confidence intervals on $\bar{z}$ can be constructed using the corresponding sample mean and variance [13, Sec.3].

Backward step. This step stems from the backward recursion technique used in *dynamic programming* which exploits the fact that the cost-to-go function at the terminal stage is precisely known, that is $\mathfrak{Q}_{T+1}(x_T) := 0$. To illustrate this further, let $\bar{x}_t = \bar{x}_t(\xi_{[t]})$, $\forall\, t = 1, \ldots, T-1$, be the trial decision obtained at the forward step and let $\check{\mathfrak{Q}}_t(\cdot)$ be the current approximation of the cost-to-go function, given by the maximum of a collection of cutting planes defined by $\mathfrak{Q}_t(x_{t-1}) = \max_{j\in J_t}\{\beta_t^{j\top} x_{t-1} + \alpha_t^j\}$, $\forall\, t = 2, \ldots, T$. At stage $t = T$ the following problem is solved:

$$\underline{Q}_T(\bar{x}_{T-1}, \xi_T) = \begin{cases} \min\limits_{x_T\in\mathbb{R}^{n_T}_+} & c_T^\top x_T \\ \text{s.t.} & A_T x_T = b_T - B_T \bar{x}_{T-1} \end{cases} \tag{3}$$

$\forall\, \xi_T = (c_T, B_T, A_T, b_T) \in \Xi_T$. Let $\bar{\pi}_T = \bar{\pi}_T(\xi_T)$ be an optimal dual solution of problem (3). Then $\alpha_T := \mathbb{E}[b_T^\top \bar{\pi}_T]$ and $\beta_T := -\mathbb{E}[B_T^\top \bar{\pi}_T] \in \partial\mathfrak{Q}_T(\bar{x}_{T-1})$ define a function $q_T(x_{T-1}) := \beta_T^\top x_{T-1} + \alpha_T = \mathfrak{Q}_T(\bar{x}_{T-1}) + \langle \beta_T, x_{T-1} - \bar{x}_{T-1}\rangle$,

satisfying $\mathfrak{Q}_T(x_{T-1}) \geq q_T(x_{T-1})$, $\forall\, x_{T-1}$, and $\mathfrak{Q}_T(\bar{x}_{T-1}) = q_T(\bar{x}_{T-1})$, i.e., $q_T(\cdot)$ is a supporting cutting plane for $\mathfrak{Q}_T(\cdot)$. This cutting plane is thereby used to update $\check{\mathfrak{Q}}_T(\cdot)$ to $\check{\mathfrak{Q}}_T(x_{T-1}) := \max\{\check{\mathfrak{Q}}_T(x_{T-1}), q_T(x_{T-1})\}$. The updated model $\check{\mathfrak{Q}}_T(\cdot)$ is then used at stage $T-1$, and we recursively solve the following problem for $t = T-1,\ldots,2$:

$$\underline{Q}_t(\bar{x}_{t-1},\xi_t) = \begin{cases} \min\limits_{x_t\in\mathbb{R}^{n_t}_+} c_t^\top x_t + \check{\mathfrak{Q}}_{t+1}(x_t) \\ \text{s.t.} \quad A_t x_t = b_t - B_t\bar{x}_{t-1} \end{cases} \equiv \begin{cases} \min\limits_{(x_t,r_{t+1})\in\mathbb{R}^{n_t}_+\times\mathbb{R}} c_t^\top x_t + r_{t+1} \\ \text{s.t.} \quad A_t x_t = b_t - B_t\bar{x}_{t-1} \\ \qquad \beta_{t+1}^{j\top} x_t + \alpha_{t+1}^j \leq r_{t+1}, \quad j\in J_{t+1}\,. \end{cases} \tag{4}$$

Similarly, let $\bar{\pi}_t = \bar{\pi}_t(\xi_t)$ and define $\bar{\rho}_t^j = \bar{\rho}_t^j(\xi_t)$ as the optimal dual multipliers associated with constraints $A_t x_t = b_t - B_t\bar{x}_{t-1}$ and $\beta_{t+1}^{j\top}x_t + \alpha_{t+1}^j \leq r_{t+1}$, respectively. Then the function $q_t(x_{t-1}) := \beta_t^\top x_{t-1} + \alpha_t = \mathbb{E}[\underline{Q}_t(\bar{x}_{t-1},\xi_t)] + \langle \beta_t, x_{t-1} - \bar{x}_{t-1}\rangle$ constructed with $\alpha_t := \mathbb{E}[b_t^\top\bar{\pi}_t + \sum_{j\in J_{t+1}}\alpha_{t+1}^j\bar{\rho}_t^j]$ and $\beta_t := -\mathbb{E}[B_t^\top\bar{\pi}_t] \in \mathbb{E}[\partial\underline{Q}_t(\bar{x}_{t-1},\xi_t)]$ is a supporting cutting plane for $\mathfrak{Q}_t(\cdot)$ which satisfies $\mathfrak{Q}_t(x_{t-1}) \geq q_t(x_{t-1})$, $\forall\, x_{t-1}$ and is added to update the collection of the supporting cutting planes at stage t: $\check{\mathfrak{Q}}_t(x_{t-1}) = \max\{\check{\mathfrak{Q}}_t(x_{t-1}), q_t(x_{t-1})\}$. This process continues until the first stage, where the solution of (4) at $t=1$ defines a lower bound $\underline{z}$ for the optimal value of (1).

3. Adaptive partition-based approach for stochastic linear programs with fixed recourse

Decomposition algorithms used for solving stochastic programs usually rely on sequences of candidate solutions (trial points in the sequel) $\bar{x}_t = \bar{x}_t(\xi_{[t]})$ for generating cutting plane approximations. It is intuitively clear that generating such approximations to $\mathfrak{Q}_t(\cdot)$'s with high precision for early iterates, likely far from an optimal solution, is surely a wasteful use of computing power — accuracy will need to be integrated adaptively when more precision is necessary in the resulting solution. Hence, as a mean to make the cut generation effort adaptive to the solution progress, a natural idea is to partition the scenario set, for a given stage, into clusters (resulting in a relaxation of the original stochastic program) and approximate the functions $\mathfrak{Q}_t(\cdot)$'s using one representative scenario for each cluster, which aggregates the information for the cluster. The partition is then refined (when necessary) dynamically during the iterative solution process to better approximate $\mathfrak{Q}_t(\cdot)$. To demonstrate this, we assume a fixed recourse structure from now on, i.e., c_t and A_t for all $t = 2,3,\ldots,T$ are assumed to be deterministic, implying that $\{(B_t^{\xi_t}, b_t^{\xi_t})\}_{\xi_t=1}^{|\Xi_t|}$ is the set of realizations used to characterize the uncertainty, $\bar{x}_t$ be the trial points, L_t the partition size and let the current partition be denoted as $\mathcal{N}_t = \{\mathcal{P}_t^\ell\}_{\ell=1}^{L_t}$ for each stage $t = 2,3,\ldots,T$. At stage $t=T$, the scenario-based subproblem for each realization $(B_T^{\xi_T}, b_T^{\xi_T}) \in \Xi_T$ and the partition-based subproblem for each cluster $\mathcal{P}_T^\ell \in \mathcal{N}_T$, $\ell = 1,2,\ldots,L_T$ compare as follows:

$$\sum_{\xi_T\in\mathcal{P}_T^\ell}\Big[\underline{Q}_T(\bar{x}_{T-1},\xi_T)\Big] \geq \underline{Q}_T(\bar{x}_{T-1},\mathcal{P}_T^\ell) := \min_{x_T}\Big\{c_T^\top x_T \mid A_T x_T = \bar{b}_T^\ell - \bar{B}_T^\ell\bar{x}_{T-1}\Big\},$$

where $\bar{b}_T^\ell := \sum_{\xi_t\in\mathcal{P}_T^\ell} b_T^{\xi_T}$, and $\bar{B}_T^\ell := \sum_{\xi_t\in\mathcal{P}_T^\ell} B_T^{\xi_T}$. It can be easily seen that $\underline{Q}_T(\bar{x}_{T-1},\mathcal{P}_T^\ell)$ is constructed by aggregating constraints and variables of $\underline{Q}_T(\bar{x}_{T-1},\xi_T)$, $\forall\xi_T\in\mathcal{P}_T^\ell$, and therefore gives a valid lower bound. Treating each cluster $\mathcal{P}_T^\ell$ as a scenario, a coarse cut $\beta_T^{j\top}x_{T-1} + \alpha_T^j \leq r_T$ can be generated in the same way that a standard *Benders* cut is generated (see the derivations after equation (3)), using the corresponding optimal dual solution for each cluster $\mathcal{P}_T^\ell$ of $\mathcal{N}_T$. When the coarse cuts do not yield any cut violation at the trial point $\bar{x}_{T-1}$ with respect to the current relaxation for the cost-to-go function, i.e., $\check{\mathfrak{Q}}_T(\bar{x}_{T-1}) \geq \beta_T^{j\top}\bar{x}_{T-1} + \alpha_T^j$, the partition $\mathcal{N}_T$ can be refined by solving the subproblem $\underline{Q}_T(\bar{x}_{T-1},\xi_T)$, $\forall\,\xi_T\in\Xi_T$, where the corresponding optimal dual vectors will guide the partition refinements. In this paper we will adopt a very simple refinement strategy which is, we isolate any two realizations from a cluster $\mathcal{P}_t$ of the partition $\mathcal{N}_t$ whenever their euclidian distance is sufficiently large. It follows that, the refined partition will only contain clusters of realizations whose corresponding optimal dual vectors are sufficiently close to each other in terms of the Euclidian distance. Several other strategies can be considered such as those described in [11]. Now, similarly, consider any other stage $t\in\{2,3,\ldots,T-1\}$, using the cutting plane approximation $\check{\mathfrak{Q}}_{t+1}(x_t) = \max_{j\in J_{t+1}}\{\alpha_{t+1}^j + \beta_{t+1}^j x_t\}$ for the cost-to-go function $\mathfrak{Q}_{t+1}(x_t)$, the scenario-based subproblem for each realization $(B_t^{\xi_t}, b_t^{\xi_t}) \in \Xi_t$ and the partition-based subproblem for each cluster $\mathcal{P}_t^\ell \in \mathcal{N}_t$, $\ell = 1,2,\ldots,L_t$ compare as follows:

$$\sum_{\xi_t\in\mathcal{P}_t^\ell}\Big[\underline{Q}_t(\bar{x}_{t-1},\xi_t)\Big] \geq \underline{Q}_t^\ell(\bar{x}_{t-1},\mathcal{P}_t^\ell).$$

Similar to the case when $t=T$, the partition refinement is guided by the optimal dual vectors $\{(\lambda_{\xi_t},\pi_{\xi_t})\}_{\xi_t\in\mathcal{P}_t^\ell}$. It can be seen from [10, Theorem 2.5] that this partition refinement rule guarantees that, after refining $\mathcal{N}_t$ into $\mathcal{N}_t' = \{\mathcal{P}_t^\ell\}_{\ell=1}^{L_t'}$, $\sum_{\ell=1}^{L_t'}\underline{Q}_{t,\ell}(\bar{x}_{t-1}) = \sum_{\forall\xi_t\in\Xi_t}\underline{Q}_t(\bar{x}_{t-1},\xi_t)$.

4. Adaptive partition-enabled preprocessing for multistage stochastic linear programs

In this section we present our proposed algorithm which builds a connection between the SDDP algorithm discussed in Section 2 and the adaptive partition-based strategy of Section 3. Considering the promising performance of the adaptive partition-based strategy in the two-stage setting presented in [10, 11], it is very tempting to conclude that this competence will be extended naturally to the multistage context. However, the effectiveness of the approach in the multistage case is hindered by the following difficulty: The convergence in the two-stage case relies heavily on the fact that one only needs to define a partition over the set of second-stage realizations, and seeks a sufficient partition that induces an optimal first-stage solution; whereas in the multistage case, the sufficiency of a partition over the set of realizations in stage t depends on the particular sample path that it takes from stage 1 to stage $t-1$. Hence, using the strategy in the multistage case will only yield a sequence of partitions $\mathcal{N} = (\mathcal{N}_1, \ldots, \mathcal{N}_T)$ which are only sufficient "locally" with respect to the exercised sample path(s) $\xi = (\xi_1, \ldots, \xi_T)$ — making it (almost) impossible to identify a sequence of partitions $\mathcal{N}$ that accommodates all $\Xi_1 \times \ldots \times \Xi_T$ possible sample paths. Nonetheless, the coarse cuts generated throughout this process remain valid cuts to the original problem. While they might be weaker cuts (because only aggregated information is utilized), one could still employ these coarse cuts for the purpose of accelerating the backward step, particularly in the early stages where exact information is not necessary for generating initial cutting plane relaxations. This has been validated in our experiment results shown in Section 5.

In order to construct such "locally" sufficient sequence of partitions, our algorithm decouples problem (1) into $T-1$ consecutive two-stage problems. Each $t = 1, \ldots, T-1$ problem will be solved using the adaptive partition-based strategy on a sample path $\xi^k = (\xi_1^k, \ldots, \xi_T^k)$ which is randomly chosen from the scenario tree. This procedure will not only yield a partition $\mathcal{N}_{t+1}^k$ corresponding to ξ^k but also warm start the process of approximating the cost-to-go function at every stage t via the generated coarse cuts during the solution process. Once all $T-1$ two-stage problems are solved, we exploit the resulting sequence of partitions $\mathcal{N}^k = (\mathcal{N}_1^k, \ldots, \mathcal{N}_T^k)$ by defining a relaxation of problem (1) on the aggregated scenarios in N^k which can be solved efficiently using SDDP – now that the scenarios are aggregated which makes the size of the scenario tree significantly smaller.

The motivation behind decoupling problem (1) into $T-1$ two-stage problems is the desire to modify the setting of the cost-to-go function approximation procedure to a framework which fits and leverages the computational efficiency of the adaptive partition-based approach, and generate partitions to which we can validly claim sufficiency.

It's worth noting that this decoupling modification is nothing but an alternative strategy for traversing the scenario tree to the one employed by the SDDP algorithm, with the additional benefit of making the cut generation effort adaptive to the solution progress. This tree traversal strategy can be seen as a "cautious" strategy, in the sense that, the policy evaluation process never goes forward down the tree unless all cuts that would be passed back from stage $t+1$ to stage t are redundant. Unlike the "quick" pass used in SDDP, where the policy is evaluated at every stage ($1 \to 2 \to \cdots \to T-1 \to T$) in the scenario tree, and cuts are passed directly back up the tree ($T \to T-1 \to \cdots \to 2 \to 1$) with no intermediate changes of direction between consecutive stages t and $t-1$. However, we still employ the quick traversal strategy in the relaxation problem induced by the sequence of partitions $\mathcal{N}^k$.

SDDP with the adaptive partition-enabled preprocessing We outline our proposed algorithm which integrates SDDP with the adaptive partition-enabled approach (see also Algorithm 1) in the following three steps:

1. *Exploration step* This step is concerned with constructing a provisional sequence of partitions $\mathcal{N}$ using the *adaptive partition-based* technique, which will then be used in the *Exploitation step*. To accomplish this, for a given scenario $\xi = (\xi_1, \ldots, \xi_T)$ we define a sequence of two-stage stochastic programs between every two consecutive stages t and $t+1$, $\forall\, t = 2, \ldots, T-1$ and then solve them using the *adaptive partition-based* approach.
2. *Exploitation step* This step is concerned with exploiting the provisional sequence of partitions $\mathcal{N}$ obtained in the *Exploration step*. The sequence $\mathcal{N} = (\mathcal{N}_1, \ldots, \mathcal{N}_T)$ induces a relaxation of the original formulation (1) under which the scenarios variables and constraints are aggregated in clusters $\mathcal{P}_t^\ell$ (see Section 3) such that $\mathcal{N}_t = \{\mathcal{P}_t^\ell\}_{\ell=1}^{L_t}$, $\forall\, t = 2, 3, \ldots, T$. This induced relaxation problem is then solved using the standard SDDP algorithm and we keep track of its lower bound progress. If the progress is relatively small for n consecutive iterations, we go back to the *Exploration step* and refine $\mathcal{N}$ as discussed in Section 3.
3. *Optimization step* While alternating between between step 1 and 2, we keep track of the size of the sequence of partitions relative to the full set of scenarios, i.e., $\sum_{t=1}^{T} |\mathcal{N}_t| / |\Xi_t|$. Whenever this ratio exceeds a certain threshold parameter m, we terminate the loop and simply revert to the original stochastic program with the full set of scenarios, which is then solved using standard SDDP. The coarse cuts obtained in step 1 and 2 are used for warm-starting the SDDP.

Algorithm 1 Adaptive Partition-enabled Preprocessing algorithm.

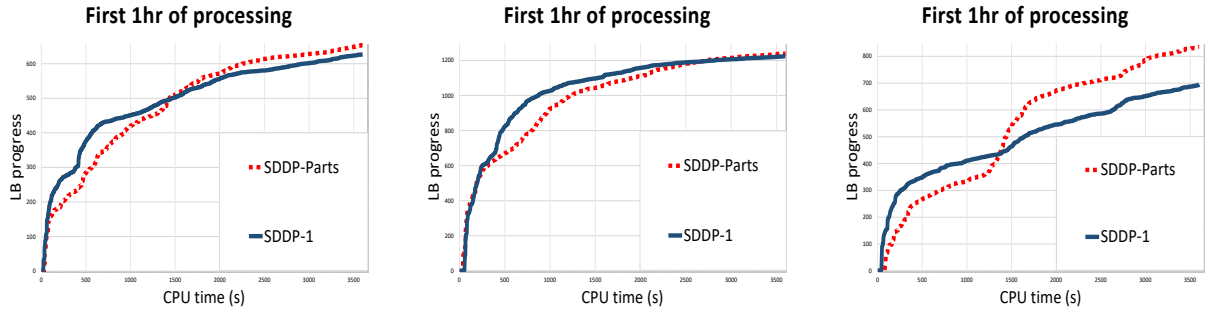
STEP 0: Initialization. Let $k = 0, \underline{z}^0 := -\infty$, $\check{\mathfrak{Q}}_t^0(\cdot) := -\infty$, $\forall\, t = 2, \ldots, T$. Choose a tolerance $\varepsilon > 0$, threshold parameter $m \in (0, 1]$, and an initial sequence of partitions $\mathcal{N}^0 = (\mathcal{N}_1^0, \ldots, \mathcal{N}_T^0)$.
STEP 1: Increment $k \to k+1$ and refine $\mathcal{N}_{t+1}^{k-1}$ by doing the following loop: $\forall\, t = 1, \ldots, T-1$ define a two-stage stochastic program on $\mathcal{N}_{t+1}^{k-1}$ where: 1^{st}-stage $= t$ and 2^{nd}-stage $= t+1$ and solve it using the adaptive partition-based approach [10, 11] to update $\mathcal{N}_{t+1}^{k-1} \to \mathcal{N}_{t+1}^{k}$ and $\mathfrak{Q}_t^{k-1}(\cdot) \to \mathfrak{Q}_t^{k}(\cdot)$.
STEP 2: If $\sum_{t=1}^{T} |\mathcal{N}_t^k| / |\Xi_t| > m$, go to STEP 3. Otherwise, define an MSLP on the sequence of partitions $\mathcal{N}^k = (\mathcal{N}_1^k, \ldots, \mathcal{N}_T^k)$ and cutting plane approximations $\mathfrak{Q}_t^k(\cdot)$, where $\mathcal{N}_t^k = \{\mathcal{P}_t^{\ell k}\}_{\ell=1}^{L_t^k}$ and the clusters $\mathcal{P}_t^{\ell k}$ are treated as scenarios. Define a parameter $n < k$ and do the following loop: While $\underline{z}^k - \underline{z}^n > \varepsilon$ apply the SDDP algorithm to improve the cutting plane approximations $\mathfrak{Q}_t^k(\cdot)\ \forall\, t = 2, \ldots, T$ and consequently update $\underline{z}^{k-1} \to \underline{z}^k$. go to STEP 1.
STEP 3: Define an MSLP on the complete set of realizations with the updated cutting plane approximations $\mathfrak{Q}_t^k(\cdot)$. Solve it using the SDDP algorithm and stop upon a termination criterion (such as a time limit).

5. Numerical experiments

In this section we present some preliminary numerical results for the performance of our proposed algorithm compared to the standard SDDP algorithm. We implemented all algorithms in *julia 0.6.2*, *JuMP 0.18.4* [15], using commercial solver *Gurobi*, version 7.0.1 [15]. All the tests are conducted on Clemson University's primary high-performance computing (HPC) resource, "big-memory" node with 2.67GHz processors, using a 494Gb memory and the number of threads is set to be 24. We benchmark the performance of each algorithm by reporting their lower bound progress at prespecified time limits. From both Table 1 and Figure 1 we can see that, our proposed algorithm outperforms

Table 1: Lower bound progress obtained by the SDDP algorithm with adaptive partition-enabled preprocessing (SDDP-Parts) and the standard SDDP algorithm. $\%\text{LB} = 100(LB_{SDDP-Parts} - LB_{SDDP})/(LB_{SDDP})$.

Time	T	25			61			97		
	$\lvert\Xi_t\rvert$	20	50	100	20	50	100	20	50	100
1 hour	%LB	4.3	1.2	19.9	5.9	1.1	4.1	5.0	0.9	-0.2
3 hours	%LB	3.9	0.3	15.2	3.3	-0.2	5.5	3.6	3.9	3.8
6 hours	%LB	3.7	1.2	10.4	2.6	-0.2	1.0	3.8	3.1	2.9
Preprocessing time in seconds		11.8	25.5	83.5	19.8	54.3	123.5	17.5	91.0	213.4



(a) $|\Xi_t| = 20$ (b) $|\Xi_t| = 50$ (c) $|\Xi_t| = 100$

Figure 1: LB progress for $T = 25$ and $|\Xi_t| = 20$, 50 and 100 over 6 hours

the standard SDDP in terms of the LB progress (in most instances). This is consistent for both small and large number of stages with different size of realizations. We also observe that the improvement is more significant when a relatively tighter time limit is adopted. The adaptive partition-enabled preprocessing is not a major factor in the overall computational time.

6. Conclusions

This work extends the computational aspects of [10, 11] in the two-stage context to the multistage setting. Although identifying a sufficient partition for optimality might be intractable due to the curse of dimensionality brought by

the large number of sample paths, we show through our numerical results that we can still make use of the adaptive partition-based approach at least during the early stages of the solution process by generating coarse cuts. This efficiency can also be exploited further by employing regularization schemes and bundle methods to stabilize the iterates during the solution process and hence accelerate the convergence. An alternative integration to the adaptive partition-based approach in multistage stochastic programs is to perform coarse cut generation and partition refiniements within the backward pass of the SDDP.

Acknowledgements

Clemson University is acknowledged for generous allotment of compute time on Palmetto cluster. The authors acknowledge partial support by the National Science Foundation [Grant CMMI 1854960]. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] de Matos, V.L., Morton, D.P. and Finardi, E.C., 2016, "Assessing policy quality in a multistage stochastic program for long-term hydrothermal scheduling," Annals of Operations Research, 253(2), 1-19,

[2] Fhoula, B., Hajji, A. and Rekik, M., 2013, "Stochastic dual dynamic programming for transportation planning under demand uncertainty," In Advanced Logistics and Transport (ICALT), May 2013 International Conference on (550-555). IEEE.

[3] Dupačová, J. and Polívka, J., 2009, "Asset-liability management for Czech pension funds using stochastic programming," Annals of Operations Research, 165(1), 5-28.

[4] van Ackooij, W., de Oliveira, W. and Song, Y., 2017, "On level regularization with normal solutions in decomposition methods for multistage stochastic programming problems (submitted for publication).

[5] Powell, W.B., 2007, "Approximate Dynamic Programming: Solving the curses of dimensionality," (Vol. 703). John Wiley & Sons.

[6] Pereira, M.V. and Pinto, L.M., 1991, "Multi-stage stochastic optimization applied to energy planning," Mathematical programming, 52(2), 359-375.

[7] Linowsky, K. and Philpott, A.B., 2005, "On the convergence of sampling-based decomposition algorithms for multistage stochastic programs," Journal of optimization theory and applications, 125(2), 349-366.

[8] Asamov, T. and Powell, W.B., 2018, "Regularized decomposition of high-dimensional multistage stochastic programs with markov uncertainty," SIAM Journal on Optimization, 28(1), 575-595.

[9] Sen, S. and Zhou, Z., 2014. Multistage stochastic decomposition: a bridge between stochastic programming and approximate dynamic programming. SIAM Journal on Optimization, 24(1), pp.127-153.

[10] Song, Y. and Luedtke, J., 2015, "An adaptive partition-based approach for solving two-stage stochastic programs with fixed recourse," SIAM Journal on Optimization, 25(3), 1344-1367.

[11] van Ackooij, W., de Oliveira, W. and Song, Y., 2017, "Adaptive Partition-Based Level Decomposition Methods for Solving Two-Stage Stochastic Programs with Fixed Recourse," INFORMS Journal on Computing, 30(1), 57-70.

[12] Kelley, Jr, J.E., 1960, "The cutting-plane method for solving convex programs," Journal of the Society for Industrial and Applied Mathematics, 8(4), 703-712.

[13] Shapiro, A., 2011, "Analysis of stochastic dual dynamic programming method," European Journal of Operational Research, 209(1), 63-72.

[14] Shapiro, A., Dentcheva, D. and Ruszczyński, A., 2009, "Lectures on stochastic programming: modeling and theory," Society for Industrial and Applied Mathematics.

[15] Dunning, I., Huchette, J. and Lubin, M., 2017, "JuMP: A modeling language for mathematical optimization," SIAM Review, 59(2), 295-320.