Adaptive Partition-based Level Decomposition Methods for Solving Two-stage Stochastic Programs with Fixed Recourse

Wim van Ackooij* Welington de Oliveira† Yongjia Song‡

February 8, 2018

Abstract

We present a computational study of several strategies to solve two-stage stochastic linear programs by integrating the adaptive partition-based approach with level decomposition. A partition-based formulation is a relaxation of the original stochastic program, obtained by aggregating variables and constraints according to a scenario partition. Partition refinements are guided by the optimal second-stage dual vectors computed at certain first-stage solutions. The proposed approaches rely on the level decomposition with on-demand accuracy to dynamically adjust partitions until an optimal solution is found. Numerical experiments on a large set of test problems including instances with up to one hundred thousand scenarios show the effectiveness of the proposed approaches.

Keywords:
Stochastic programming; scenario reduction; level decomposition

1 Introduction

In this paper we are concerned with numerically solving two-stage stochastic linear programming problems with very large sets of scenarios. In such problems, we explicitly model the fact that a first decision needs to be taken prior to observing uncertainty and then a potential recourse action can act once this uncertainty has been revealed. We are concerned with the situation known as fixed recourse, implying that the duals of all second-stage linear problems share the same feasible set. From an abstract viewpoint two-stage stochastic programming can be seen as a special case of non-linear non-smooth convex optimization by considering the second-stage expected cost as a mapping, called $\mathcal{Q}$ in the sequel, of the first-stage variables. The computational challenge then lies in computing this map efficiently, which is immediately related to the number of considered scenarios.

As a mean to solve the problem faster, a natural idea is to partition the scenario set into clusters, and approximate function $\mathcal{Q}$ using one representative scenario for each cluster, which "aggregates" the information in a cluster of scenarios. The idea of using aggregation for solving stochastic programs has a long history. [2] studies aggregation techniques to obtain optimality bounds. For multistage stochastic linear programs with a general probability distribution, [38] studies aggregation and disaggregation with respect to σ-algebras of the underlying probability space. Another concept of aggregation is proposed in [29] for two-stage stochastic programs, which considers constraints aggregation across different scenarios as well as constraints within each scenario. An aggregation procedure based on clustering of scenarios is employed in [18]. A practical implementation of a similar idea wherein scenarios are quantized can be found in [33]. Important contributions on the subject of selecting smaller set of scenarios have been done by using probability metrics [16, 27], in particular the Wasserstein distance [10]. The idea is to approximate the problem by a smaller problem defined by a fixed and small partition. The partition is obtained by applying an algorithm based on probability metrics to the set of scenarios. [26] and [7] incorporate the second-stage costs

*EDF R&D, OSIRIS, France, wim.van.ackooij@gmail.com
†Universidade do Estado do Rio de Janeiro, Brazil, welington@ime.uerj.br
‡Virginia Commonwealth University, USA, ysong3@vcu.edu

into the distance function used to define the Wasserstein metric, and show empirically that partitions with better quality could be obtained. The idea of partitions is also exploited in [30] in order to derive upper and lower bounds on the optimal value of two-stage stochastic programs. However, computing these bounds essentially requires solving as many problems as the number of partitions of a given cardinality that can be found. These bounds are quite general as they do not require a fixed recourse assumption and binary variables can be considered in the second-stage problems, but can be computationally challenging to obtain. [5] combine progressive hedging and partitioning scenarios in order to solve a set of smaller two-stage problems related to the considered subsets of scenarios. However, the partition is chosen a priori, although several strategies have been exploited to help make a good choice.

A second idea to solve the problem faster is to simply compute $\mathcal{Q}$ "inexactly". We begin here by noting that the typical algorithms will generate a sequence of candidate solutions (called trial points in the sequel) converging to the optimal solution. It becomes intuitively clear that computing $\mathcal{Q}$ 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 eventually if one is to assert optimality of the resulting solution. These ideas have been made theoretically precise in the on-demand accuracy framework [6]. The assumption of fixed recourse can be immediately put to use, as any dual feasible solution will allow us to define a valid cutting plane for the mapping $\mathcal{Q}$. In particular, whenever a fraction of $\mathcal{Q}$ has been computed and it becomes apparent that the current trial point is not optimal, it may be of interest to skip computing the remaining part of $\mathcal{Q}$. We recall here that $\mathcal{Q}$ is the expected value over a large set of scenarios, each of which involves solving a linear program. This feature has been exploited in a series of publications on on-demand accuracy. Using such inexact computations combined with stabilization methods has been shown to be quite promising. One particular method of interest as stabilization is concerned is the level bundle method [12, 21, 34]. The level bundle method employing inexact oracles with on-demand accuracy has been applied to two-stage stochastic programs in [6], [37], and [13]. In the stochastic programming context, the level bundle method is also known as *level decomposition*, see for instance [37]. The latter work presents an extensive computational study by analyzing several algorithms for two-stage stochastic linear programs, including the extended formulation, single- and multi-cut variants of the classical L-shaped method [36] and level bundle methods.

1.1 Formal problem statement

Let us make precise the problem of interest. To this end, let $X \subseteq \mathbb{R}^{n_1}_+$ be a polyhedral set, $c \in \mathbb{R}^{n_1}$ be a cost vector, and N be a finite set of scenarios $\{\xi_1, ..., \xi_{|N|}\}$, the problem is given by

$$\min_{x \in X} \; c^\mathsf{T} x + \mathcal{Q}(x)\,, \tag{1}$$

where $\mathcal{Q} : \mathbb{R}^{n_1} \to \mathbb{R} \cup \{+\infty\}$ is the recourse function defined by

$$\mathcal{Q}(x) := \sum_{i \in N} p_i Q(x; \xi_i)\,, \quad \text{with} \quad Q(x; \xi_i) := \min_{y \in \mathbb{R}^{n_2}_+} \left\{ q^\mathsf{T} y \mid T_i x + W y \geq h_i \right\}. \tag{2}$$

Each scenario $i \in N$ occurs with probability $p_i > 0$, and is associated with an outcome $\xi_i = (h_i, T_i)$, where $h_i \in \mathbb{R}^{m_2}$ and $T_i \in \mathbb{R}^{m_2 \times n_1}$. We assume that the cost vector $q \in \mathbb{R}^{n_2}$ and the recourse matrix $W \in \mathbb{R}^{m_2 \times n_2}$ are fixed for all scenarios. Given this assumption, the feasible set of the dual problem to (2) is independent of the scenario:

$$Q(x; \xi_i) = \max_{\lambda \in \mathbb{R}^{m_2}_+} \left\{ (h_i - T_i x)^\mathsf{T} \lambda \mid W^\mathsf{T} \lambda \leq q \right\}. \tag{3}$$

This property, known as fixed recourse, has been exploited in [6, 17, 35, 37], to develop efficient methods to solve (1). The key is that any feasible solution λ of (3) provides the useful inequality $(h_i - T_i x)^\mathsf{T} \lambda \leq Q(x; \xi_i), \forall i \in N$. As will be shown in Sections 2 and 3, inequalities of this type are the working horses of the proposed approaches for efficiently solving problem (1).

1.2 Contributions of the paper

Instead of applying scenario clustering and scenario reduction techniques to the original set of scenarios N in a static manner, we propose to update the scenario partitions by applying these techniques dynamically

during the iterative solution process to better approximate $\mathcal{Q}$. We aim to devise an exact solution approach based on the adaptive scenario partitions. In particular, these scenario partitions will be updated according to the second-stage dual solutions $\{\lambda_i\}_{i\in N}$ corresponding to the intermediate solutions obtained.

The idea of adaptive partitions has already been exploited in the literature such as [11] and [32]. In [32], the authors propose several schemes to define and update partitions yielding a finitely converging algorithm to solve problems of type (1). The algorithm in [32] solves a sequence of smaller problems based on scenario partitions, which are referred to as the "partition-based master problems". The scenario partitions are refined according to the optimal second-stage dual solutions of the scenario subproblems. After a partition refinement, the partition-based master problem gives a tighter relaxation of the original stochastic program. In addition to partition refinements, it is also possible to merge some scenario clusters back together in a partition, without weakening the corresponding relaxation bound, thus allowing the partition size to be kept manageable. In this work we extend [32] by integrating this partition-based solution framework with decomposition algorithms of the bundle method family. In contrast to [32], we propose an algorithm that does not require solving the partition-based master problems to optimality. Instead, the algorithm dynamically updates the partition along the optimization process, which makes it possible to avoid spending unnecessary time on solving a partition-based master problem with an "unpromising" partition. This feature, useful to speed up the optimization process, is made possible thanks to the on-demand accuracy concept proposed in [6]. Here in order to position our work with respect to [37] on level bundle methods with on-demand accuracy, our contribution is the integration of level decomposition with the adaptive partition-based framework.

The contributions of this paper can be summarized as follows.

- First, the proposed approaches integrate the adaptive partition-based scheme in [32] with level bundle methods (with on-demand accuracy), which provides more flexibility in choosing the trial points to perform partition refinement. This additional flexibility may help to avoid spending too much computational effort on an unpromising partition.
- Second, we present empirical evidence that the proposed algorithms are effective compared to existing solution approaches through an extensive computational study.
- Third, we show that for general two-stage stochastic linear programs with fixed recourse, there exists a particular partition whose corresponding partition-based master problem gives an optimal solution to the original stochastic program, and the size of this partition is *independent* of number of scenarios. This result generalizes the one for the special case on simple recourse proposed in [32].

As a subproduct of our work, we establish a link between the following works on two-stage stochastic programs: [10, 18, 26] on scenario reduction; [6, 7, 37] on inexact bundle methods; and [32] on the adaptive partition-based approach.

1.3 Organization of the paper

The rest of the paper is organized as follows. Section 2 reviews some useful properties of the recourse function, provides a theoretical result on the existence of a small *sufficient* partition, and studies some strategies to obtain good-quality partitions. The proposed algorithms along with their convergence analysis are presented in Section 3. An extensive computational analysis is presented in Section 4, where the proposed algorithms with various partition refinement schemes are applied to solve a large number of test instances available in the literature of two-stage stochastic linear programs.

2 Partitions, clustering and approximations of the recourse function

This section is split into the following three subsections. Section 2.1 reviews some properties of the recourse function and presents the considered aggregated problem, approximating (1); Section 2.2 shows the theoretical contribution of this work on the existence of a partition from which an optimal solution to the original stochastic program can be obtained, whose size is independent of the number of scenarios; Section 2.3 presents three techniques for choosing partitions. We start by giving a formal definition of a partition.

A partition $\mathcal{P} = \{P_1, P_2, \cdots, P_L\}$ of the scenario set N is a collection of nonempty subsets of scenarios such that $P_1 \cup P_2 \cup \cdots \cup P_L = N$, and $P_i \cap P_j = \emptyset, \forall i, j \in \{1, 2, \ldots, L\}, i \neq j$. Given a partition $\mathcal{P}$, the recourse function in (2) is alternatively written by $\mathcal{Q}(x) = \sum_{i \in N} p_i Q(x; \xi_i) = \sum_{j=1}^{L} \sum_{i \in P_j} p_i Q(x; \xi_i)$. For each $j = 1, \ldots, L$, let $\bar{\xi}_j$ be a scenario chosen to represent all scenarios in P_j. Associating $\bar{\xi}_j$ with a weight $\pi_j = \sum_{i \in P_j} p_i$, the recourse function $\mathcal{Q}(x)$ can be approximated by

$$\sum_{j=1}^{L} \sum_{i \in P_j} p_i Q(x; \bar{\xi}_j) \quad = \quad \sum_{j=1}^{L} Q(x; \bar{\xi}_j) \sum_{i \in P_j} p_i \quad = \quad \sum_{j=1}^{L} \pi_j Q(x; \bar{\xi}_j) \,. \tag{4}$$

Many different representative scenarios $\bar{\xi}_j$ can be chosen for the same partition $\mathcal{P}$, and therefore many approximating functions of $\mathcal{Q}$ are possible for the same $\mathcal{P}$. We emphasize that the approximating function $\sum_{j=1}^{L} \pi_j Q(x; \bar{\xi}_j)$ can underestimate or overestimate the mapping $\mathcal{Q}$. From an algorithmic point of view, obtaining lower approximations is more convenient: the (lower) inexact cutting-plane never cuts off the solution set [6]. If the recourse matrix W and the cost vector q are fixed, a lower approximating function $\sum_{j=1}^{L} \pi_j Q(x; \bar{\xi}_j)$ of $\mathcal{Q}$ can be easily obtained, as shown below.

2.1 Subgradients and lower approximations of $\mathcal{Q}$

As shown in [31, Chap. 2], the recourse function $\mathcal{Q}$ is convex and bounded at a given x if the second-stage problem (2) has nonempty primal and dual feasible sets for each scenario $i \in N$. As a result, whenever $\mathcal{D}\mathrm{om}(\mathcal{Q})$ is nonempty, the function is polyhedral and subdifferentiable, with $\partial \mathcal{Q}(x) = \sum_{i \in N} p_i \partial Q(x; \xi_i)$ [31, Proposition 2.3]. Moreover, [31, Proposition 2.2] show that for any given $x \in X$ and ξ_i such that $Q(x; \xi_i)$ is finite, function $Q(\cdot; \xi_i)$ is subdifferentiable at x, with $\partial Q(x; \xi_i) = -T_i^{\mathsf{T}} \mathcal{D}(x; \xi_i)$, where $\mathcal{D}(x; \xi_i) = \arg\max\{(h_i - T_i x)^{\mathsf{T}} \lambda \mid W^{\mathsf{T}} \lambda \leq q, \lambda \in \mathbb{R}^{m_2}_+\}$ is the set of solutions to the dual problem (3).
Let λ be an arbitrary feasible point for (3). Then we have, for all $i \in N$ and $x \in X$,

$$Q(x; \xi_i) \geq (h_i - T_i x)^{\mathsf{T}} \lambda = \alpha_i(\lambda) x + \beta_i(\lambda) \;\text{ with }\; \left\{ \begin{array}{l} \alpha_i(\lambda) := -\lambda^{\mathsf{T}} T_i \\ \beta_i(\lambda) := h_i^{\mathsf{T}} \lambda \,. \end{array} \right. \tag{5}$$

Moreover, by defining $\varepsilon := \sum_{i \in N} [Q(x; \xi_i) - (h_i - T_i x)^{\mathsf{T}} \lambda] \geq 0$ one can show that

$$\mathcal{Q}(z) \geq \sum_{i \in N} p_i \alpha_i(\lambda) z + \sum_{i \in N} p_i \beta_i(\lambda) = \mathcal{Q}(x) + \sum_{i \in N} p_i \alpha_i(\lambda)(z - x) - \varepsilon \,,$$

for all $z \in X$, i.e., $\sum_{i=1}^{N} p_i \alpha_i(\lambda) \in \partial_\varepsilon \mathcal{Q}(x)$.

In the remainder of this work we assume for convenience of presentation that problem (1) has *relatively complete recourse*, that is, $X \subseteq \mathcal{D}\mathrm{om}(\mathcal{Q})$. Our results can be easily extended to the more general case when $X \setminus \mathcal{D}\mathrm{om}(\mathcal{Q}) \neq \emptyset$ by taking into account feasibility cuts.

Representative scenarios as the cluster average. Let $\pi_j = \sum_{i \in P_j} p_i > 0$ and a feasible dual solution λ be given. It follows from (5) that

$$\sum_{i \in P_j} \frac{p_i}{\pi_j} Q(x; \xi_i) \geq \sum_{i \in P_j} \frac{p_i}{\pi_j} \left(\alpha_i(\lambda) x + \beta_i(\lambda)\right) = \left(\sum_{i \in P_j} \frac{p_i}{\pi_j} h_i - \sum_{i \in P_j} \frac{p_i}{\pi_j} T_i x \right)^{\mathsf{T}} \lambda \,, \quad \forall\, x \in X \,. \tag{6}$$

In order to obtain the tightest possible lower approximation for $\sum_{i \in P_j} \frac{p_i}{\pi_j} Q(x; \xi_i)$ given by a single feasible dual solution λ, we consider an optimal solution $\lambda_j^{\mathtt{av}}$ of the following *aggregate* linear program:

$$\max_{\lambda \in \mathbb{R}^{m_2}_+} \left\{ (h_j^{\mathtt{av}} - T_j^{\mathtt{av}} x)^{\mathsf{T}} \lambda \mid W^{\mathsf{T}} \lambda \leq q \right\} \quad \text{with} \quad \left\{ \begin{array}{ll} h_j^{\mathtt{av}} := \sum_{i \in P_j} \frac{p_i}{\pi_j} h_i & \\ T_j^{\mathtt{av}} := \sum_{i \in P_j} \frac{p_i}{\pi_j} T_i & \forall\, j = 1, \ldots, L \,. \end{array} \right. \tag{7}$$

Let $\xi_j^{\mathtt{av}}$ be defined as $\xi_j^{\mathtt{av}} := (h_j^{\mathtt{av}}, T_j^{\mathtt{av}}) = \sum_{i \in P_j} \frac{p_i}{\pi_j} \xi_i$, and following (3), define the optimal value of (7) as $Q(x; \xi_j^{\mathtt{av}})$. We have thus shown that

$$\sum_{i \in P_j} \frac{p_i}{\pi_j} Q(x; \xi_i) \geq Q(x; \xi_j^{\mathtt{av}}), \quad \text{and therefore,} \tag{8}$$

$$\mathcal{Q}(x) = \sum_{j=1}^{L} \pi_j \sum_{i \in P_j} \frac{p_i}{\pi_j} Q(x; \xi_i) \geq \sum_{j=1}^{L} \pi_j Q(x; \xi_j^{\mathtt{av}}) =: \mathcal{Q}^{\mathtt{av}}(x), \ \forall x \in X. \tag{9}$$

Note that the above inequality may not hold if either the recourse matrix W or the cost vector q varies along different scenarios. Once (7) is solved for all $j = 1, \ldots, L$ and given $x \in X$, we can proceed as above and compute a subgradient $\alpha^{\mathtt{av}}$ of the aggregate function $\mathcal{Q}^{\mathtt{av}}$ at x. It turns out that $\alpha^{\mathtt{av}}$ is also an approximate subgradient of the recourse function $\mathcal{Q}$ at x (a property that is exploited by the algorithms given below). We formalize this result with the following lemma, whose proof follows from simple manipulations of (6) and (9) (see also [23, Chap. 2]).

Lemma 2.1 *Let $x \in X$ be given and $\lambda_j^{\mathtt{av}}$ be a solution of (7) for $j = 1, \ldots, L$. Let $\alpha^{\mathtt{av}}$ and $\beta^{\mathtt{av}}$ be defined by $\alpha^{\mathtt{av}} := -\sum_{j=1}^{L} \pi_j (\lambda_j^{\mathtt{av}})^{\mathsf{T}} T_j^{\mathtt{av}}$ and $\beta^{\mathtt{av}} := \sum_{j=1}^{L} \pi_j (h_j^{\mathtt{av}})^{\mathsf{T}} \lambda_j^{\mathtt{av}}$. Then $\alpha^{\mathtt{av}} \in \partial_\varepsilon \mathcal{Q}(x)$, with $\varepsilon = \mathcal{Q}(x) - \mathcal{Q}^{\mathtt{av}}(x) \geq 0$. Moreover, it follows that*

$$\mathcal{Q}(z) \geq \alpha^{\mathtt{av}} z + \beta^{\mathtt{av}} = \mathcal{Q}^{\mathtt{av}}(x) + \alpha^{\mathtt{av}}(z - x) - \varepsilon, \ \forall z \in X.$$

Suppose that for a given $x \in X$ the approximating function $\mathcal{Q}^{\mathtt{av}}$ coincides with $\mathcal{Q}$. Then $\varepsilon = \mathcal{Q}(x) - \mathcal{Q}^{\mathtt{av}}(x) = 0$ and $\alpha^{\mathtt{av}}$ is a true subgradient of $\mathcal{Q}$ at x. Therefore, the quality of the approximating function $\mathcal{Q}^{\mathtt{av}}$ (under the assumptions of fixed recourse W and cost q) depends on how different the second-stage dual solutions are in each scenario cluster in the partition. It thus makes sense to construct a partition $\mathcal{P}$ composed of clusters P_j that contain (nearly) identical dual solutions and approximate problem (1) by:

$$\min_{x \in X} \ c^{\mathsf{T}} x + \mathcal{Q}^{\mathtt{av}}(x), \ \text{where } \mathcal{Q}^{\mathtt{av}}(x) = \sum_{j=1}^{L} \pi_j Q(x; \xi_j^{\mathtt{av}}) \,. \tag{10}$$

This is the main motivation behind the partition scheme of [32], which is extended below.

2.2 Existence of a small sufficient partition

We first show that the partition-based approach proposed in [32] is related to formulation (10) if probabilities p_i are identical for all $i \in N$. In this situation the weight π_j is given by $\frac{|P_j|}{|N|}$, and the average scenario $\xi_j^{\mathtt{av}}$ becomes $\xi_j^{\mathtt{av}} = \frac{1}{|P_j|} \xi_j^{\mathtt{sum}}$, where $\xi_j^{\mathtt{sum}} := \sum_{i \in P_j} \xi_i$, and therefore

$$\frac{1}{|P_j|} Q(x; \xi_j^{\mathtt{sum}}) = \max_{\lambda \in \mathbb{R}_+^{m_2}} \left\{ \left(\frac{h_j^{\mathtt{sum}}}{|P_j|} - \frac{T_j^{\mathtt{sum}}}{|P_j|} x \right)^{\mathsf{T}} \lambda \,\middle|\, W^{\mathsf{T}} \lambda \leq q \right\} = Q(x; \xi_j^{\mathtt{av}}) \,.$$

Moreover, the definition of $\mathcal{Q}^{\mathtt{av}}$ in (10) yields the relation $\mathcal{Q}^{\mathtt{av}}(x) = \sum_{j=1}^{L} \pi_j Q(x; \xi_j^{\mathtt{av}}) = \frac{1}{|N|} \sum_{j=1}^{L} Q(x; \xi_j^{\mathtt{sum}})$, and problem (10) is, for the same partition $\mathcal{P}$, equivalent to

$$\min_{x \in X} c^{\mathsf{T}} x + \mathcal{Q}^{\mathtt{sum}}(x), \quad \text{with} \quad \mathcal{Q}^{\mathtt{sum}}(x) := \frac{1}{|N|} \sum_{j=1}^{L} Q(x; \xi_j^{\mathtt{sum}}) \,. \tag{11}$$

We recall that (11) is referred to as the *partition-based master problem* in [32], and $\mathcal{P}$ is called a *sufficient partition* if the partition-based problem (11) gives the same optimal objective value as the original stochastic program (1). Moreover, for the particular case of simple recourse and identical probabilities $p_i = 1/|N|$, [32] have shown that there exists a sufficient partition of a small size, which is independent of the number of

scenarios. We extend this result to any two-stage stochastic linear program with fixed recourse. Assume that the first-stage feasible set $X := \{x \in \mathbb{R}^{n_1}_+ \mid Ax = b\}$. Consider the dual formulation of (1):

$$\max_{\lambda,\eta} b^\mathsf{T}\eta + \sum_{i\in N} p_i h_i^\mathsf{T}\lambda^i \tag{12a}$$

$$\text{s.t. } A^\mathsf{T}\eta + \sum_{i\in N} p_i T_i^\mathsf{T}\lambda^i \le c \tag{12b}$$

$$W^\mathsf{T}\lambda^i \le q\,, \lambda^i \in \mathbb{R}^{m_2}_+,\ \forall i \in N, \eta \text{ free}. \tag{12c}$$

Let $D := \{\lambda \in \mathbb{R}^{m_2}_+ \mid W^\mathsf{T}\lambda \le q\}$ be the dual feasible set of the second-stage problem. Let $E = \{\hat{\lambda}^l\}_{l=1}^{|E|}$ be the set of all the extreme points of D. Given an extreme point optimal solution of (12a), (η^*, λ^*), let $K(l) := \{i \in N \mid (\lambda^*)^i = \hat{\lambda}^l\}$, i.e., the set of scenarios whose corresponding λ^* is identical to the l-th extreme point in set E, and let $K_2 = N \setminus \bigcup_{l=1}^{|E|} K(l)$. Based on Proposition 2.3 of [32], partition $\mathcal{P} := \{K(1), K(2), \ldots, K(|E|), \{i\}_{i\in K_2}\}$ is a sufficient partition. Theorem 2.1 shows that $|K_2| \le n_1 - m_1$, so that the size of partition $\mathcal{P}$ is $|\mathcal{P}| \le n_1 - m_1 + |E|$, which is a number that is independent of number of scenarios N.

Theorem 2.1 *Assume that the second-stage feasible set $D := \{\lambda \in \mathbb{R}^{m_2}_+ \mid W^\mathsf{T}\lambda \le q\}$ is nondegenerate, i.e., all extreme points $\{\hat{\lambda}^l\}_{l=1}^{|E|}$ satisfy exactly m_2 inequalities of D as equations (which is satisfied, e.g., if the LICQ condition holds). Let $(x^*, \lambda_1^*, ..., \lambda_N^*)$ be an optimal solution to* (1)*, let $K(l) := \{i \in N \mid (\lambda^*)^i = \hat{\lambda}^l\}$, and let $K_2 = N \setminus \bigcup_{l=1}^{|E|} K(l)$. Then $\mathcal{P} := \{K(1), K(2), \ldots, K(|E|), \{i\}_{i\in K_2}\}$ is a sufficient partition and $|K_2| \le n_1 - m_1$.*

Proof. Let us first note that by [1, Theorem 3.4.1], polyhedron D can be decomposed as $D = D^\mathsf{c} + D^\infty$, where $D^\infty = \left\{\lambda \in \mathbb{R}^{m_2}_+ \mid W^\mathsf{T}\lambda \le 0\right\}$ is the recession cone of D and D^c is a bounded polyhedron. It is clear that, E, the set of extreme points of D, is also the set of extreme points of D^c. Moreover, at x^*, we may assume without loss of generality that any λ_j^* decomposes as $\lambda_j^* = \mu_j^* + 0$, with $\mu_j^* \in D^\mathsf{c}$. Indeed by (3) if a nonzero element of the recession cone is needed, along which $(h_i - T_i x)$ admits a positive component, then $Q(x^*, \xi_i) = \infty$, which contradicts the fact that $x^* \in \mathcal{D}\text{om}(Q)$. To the contrary, if $(h_i - T_i x) \le 0$, $0 \in D^\infty$ by the optimality of x^*.

Following the assumption, any extreme point of D satisfies exactly m_2 equations. Let $(\eta^*, \lambda_1^*, ..., \lambda_{|N|}^*)$ be an extreme point optimal dual solution to (1) corresponding to $(x^*, \lambda_1^*, ..., \lambda_{|N|}^*)$. Then there are at least $m_1 + Nm_2$ active constraints in (12) by (η^*, λ^*). Constraints (12b) can contribute at most n_1 active constraints, so there are at least $m_1 + Nm_2 - n_1$ active constraints from system $\{(\lambda^1, \ldots, \lambda^{|N|}) \mid W^\mathsf{T}\lambda^i \le q, \lambda^i \in \mathbb{R}^{m_2}_+,\ \forall i = 1, 2, \ldots, |N|\}$. Also, we have at most $m_2 \times |E|$ equations from sets $K(1), K(2), \ldots, K(|E|)$ (since some of these sets may be empty), so we need at least $m_1 + (N - |E|)m_2 - n_1 = m_1 + |K_2|m_2 - n_1$ equations from set K_2. Since points in set K_2 are non-extreme points of D, for each of these points, there are at most $m_2 - 1$ active constraints from D, which altogether contribute $|K_2|(m_2 - 1)$ equations. Therefore, we have:

$$|K_2|(m_2 - 1) \ge m_1 + |K_2|m_2 - n_1,$$

and $|K_2| \le n_1 - m_1$ follows. □

Theorem 2.1 justifies the adaptive partition-based framework for solving two-stage stochastic programs with fixed recourse. It shows the existence of a sufficient partition whose size is independent of number of scenarios. Therefore, when $n_1 - m_1 + |E| \ll |N|$, the large-scale problem (1) can potentially be solved by the much smaller partition-based problem (11). Computational results shown in Section 4 empirically verify the effectiveness of the partition-based framework.

2.3 Alternative partition refinement strategies

Although the existence of a sufficient partition is ensured by Theorem 2.1, determining such partition is not an easy task in general. In order to have a reasonable partition $\mathcal{P} = \{P_1, \ldots, P_L\}$, [32] suggest to aggregate similar dual solutions:

$$P_j := \{j_1, j_2, \ldots, j_k\} \quad \text{such that} \quad \|\lambda_{j_i}^k - \lambda_{j_l}^k\| \le \delta,\ \ \forall\, i, l \in N \tag{13}$$

where $\delta > 0$ is a given tolerance parameter and λ_i^k is a dual optimal solution of (3) with x replaced by the current point x^k of an iterative process. Depending on the number of extreme points $|E|$, this "absolute" strategy (13) may fail to yield a small number of components after a refinement. We therefore examine two alternative approaches, one based on scenario clustering and another employing some ideas from scenario reduction techniques.

Scenario clustering based on dual solutions. Let $\lambda_j^{\mathtt{av}}$ be the average of dual solutions λ_i^k associated to the component P_j of partition $\mathcal{P}$. One possibility to determine a small partition $\mathcal{P}$ consists in finding a (local) solution of the following combinatorial problem

$$\min_{\substack{P_1,\ldots,P_{L_k} \\ P_j \cap P_j = \emptyset \ \forall j \neq l}} \sum_{j=1}^{L_k} \sum_{i \in P_j} \|\lambda_i^k - \lambda_j^{\mathtt{av}}\|^2 \ \text{ s.t. } \ P_1 \cup P_2 \cup \ldots \cup P_{L_k} = N, \tag{14}$$

where L_k is the size of a new partition, which can vary along the iterative process. A local solution to the above problem can be obtained by applying the well-known K-means algorithm of [24].

Scenario reduction based on dual solutions. Another manner for defining partitions is to employ the scenario reduction technique proposed in [10], but with distance d_{ij} between scenarios ξ_i and ξ_j replaced by distances between dual solutions λ_i^k and λ_j^k (for instance $d_{ij} = \|\lambda_i^k - \lambda_j^k\|$). Let $I \subseteq \{1, 2, \ldots, |N|\}$ be a (local) solution to the following combinatorial optimization problem

$$\min_{I} \sum_{i \in I} p_i \min_{j \in \{1,\ldots,|N|\} \setminus I} d_{ij} \ \text{ s.t. } I \subseteq \{1, \ldots, |N|\}, \ |I| = |N| - L_k \,, \tag{15}$$

that can be obtained by one of the heuristics of [15]. A partition $\mathcal{P} = \{P_1, \ldots, P_{L_k}\}$ is thus constructed by assigning to the cluster P_j all the scenario indices $i \in I$ such that λ_i^k is the dual solution nearest to the j^{th} solution $\lambda_{s_j}^k$ with $s_j \in \{1, \ldots, |N|\} \setminus I$.

These three alternatives (the absolute strategy (13), scenario clustering, and scenario reduction) are numerically assessed in Section 4.

3 Stabilized cutting-plane algorithms with adaptive partitions

In this section we combine the adaptive partition-based framework with level decomposition for solving two-stage stochastic linear programs. Our analysis relies on two types of linearizations to approximate the costly function $\mathcal{Q}(x)$ in (1): (a) fine cuts (exact oracle) and (b) coarse cuts (inexact oracle).

3.1 Ingredients of the approach: cutting-plane models, coarse and fine cuts

Let $k \in \mathbb{Z}_+$ be an iteration counter, $\ell \in \mathbb{Z}_+$ be a partition counter, and let $\mathcal{P}^\ell$ be the ℓ-th partition whose size is $L_\ell \leq |N|$. Given a partition $\mathcal{P}^\ell = \{P_1^\ell, \ldots, P_{L_\ell}^\ell\}$, we define as before that $\pi_j = \sum_{i \in P_j^\ell} p_i$ and $\xi_j^{\mathtt{av}} = \sum_{i \in P_j^\ell} \frac{p_i}{\pi_j} \xi_i$, and the resulting average approximation function is $\mathcal{Q}_\ell^{\mathtt{av}}(x) = \sum_{j=1}^{L_\ell} \pi_j Q(x; \xi_j^{\mathtt{av}})$.

Coarse cuts. Given an iterate $x^k \in X$, a linearization of $\mathcal{Q}_\ell^{\mathtt{av}}$ at point x^k defines the k-th coarse linearization for $\mathcal{Q}$ at x^k:

$$\mathcal{Q}(x) \geq \mathcal{Q}_\ell^{\mathtt{av}}(x) \geq \alpha_k^{\mathtt{av}} \, x + \beta_k^{\mathtt{av}}, \quad \forall \, x \in X, \quad \text{with} \quad \left\{ \begin{array}{ll} \alpha_k^{\mathtt{av}} & = -\sum_{j=1}^{L_\ell} \pi_j (\lambda_j^{\mathtt{av}})^{\mathsf{T}} T_j^{\mathtt{av}} \\ \beta_k^{\mathtt{av}} & = \sum_{j=1}^{L_\ell} \pi_j (h_j^{\mathtt{av}})^{\mathsf{T}} \lambda_j^{\mathtt{av}} \,, \end{array} \right. \tag{16}$$

where $\lambda_j^{\mathtt{av}}$ is an optimal solution to problem (3) with $(x; \xi_i)$ replaced by $(x^k; \xi_j^{\mathtt{av}})$. Lemma 2.1 shows that the above inequality is valid. Let J_k^c be the index set that gathers all the coarse cuts up to iteration k.

Fine cuts. A fine cut for function $\mathcal{Q}$ is computed at iteration k if all the subproblems (3), with x replaced by x^k, are solved to optimality. Let the corresponding optimal dual solutions be λ_i for all $i \in N$:

$$\mathcal{Q}(x) \geq \alpha_k\, x + \beta_k, \quad \forall\, x \in X, \quad \text{with} \quad \left\{ \begin{array}{ll} \alpha_k & = -\sum_{i \in N} p_i \lambda_i^{\mathsf{T}} T_i \\ \beta_k & = \sum_{i \in N} p_i h_i^{\mathsf{T}} \lambda_i\,, \end{array} \right. \tag{17}$$

Note that fine cuts are more expensive to compute than the coarse ones: in order to compute a fine cut, $|N|$ second-stage problems need to be solved, whereas a coarse cut requires the solution of only L_ℓ ($\leq |N|$) second-stage subproblems. Let J_k^f be the index set that gathers all the fine cuts up to iteration k.

Cutting-plane approximation. It follows from the convexity of $\mathcal{Q}$, $\mathcal{Q}_\ell^{\mathtt{av}}$, and the definition of the coarse and fine cuts that the following cutting-plane model gives an underestimate of $\mathcal{Q}(x)$:

$$\check{\mathcal{Q}}_k(x) := \max \left\{ \max_{l \in J_k^c} \{\alpha_l^{\mathtt{av}}\, x + \beta_l^{\mathtt{av}}\},\, \max_{l \in J_k^f} \{\alpha_l\, x + \beta_l\} \right\}, \tag{18}$$

i.e., $\check{\mathcal{Q}}_k(x) \leq \mathcal{Q}(x)$, $\forall x \in X$. Therefore, for any given parameter $f_{\mathtt{lev}}^k \in (-\infty, \infty)$ the following relationship holds: $\{x \in X : c^{\mathsf{T}} x + \mathcal{Q}(x) \leq f_{\mathtt{lev}}^k\} \subset \{x \in X : c^{\mathsf{T}} x + \check{\mathcal{Q}}_k(x) \leq f_{\mathtt{lev}}^k\}$. As a consequence, if the right-hand side level set is empty then the level parameter $f_{\mathtt{lev}}^k$ is a lower bound for the optimal value of (1). The algorithms suggested in this paper will work with a parameter $f_{\mathtt{low}}^k$, a lower estimate of the optimal value of problem (1). The algorithms, which are variants of the level bundle methods originally proposed in [21], will update this parameter thanks to the above described mechanism. The following two steps are the essential working horses of the framework:

- If $\{x \in X : c^{\mathsf{T}} x + \check{\mathcal{Q}}_k(x) \leq f_{\mathtt{lev}}^k\} = \emptyset$, then $f_{\mathtt{low}}^{k+1}$ is set to $f_{\mathtt{lev}}^k$;
- If $\{x \in X : c^{\mathsf{T}} x + \check{\mathcal{Q}}_k(x) \leq f_{\mathtt{lev}}^k\} \neq \emptyset$, then $f_{\mathtt{low}}^{k+1}$ receives $f_{\mathtt{low}}^k$ and x^{k+1} is the projection of the stability center $\hat{x}^k$ onto this level set:

$$x^{k+1} = \left\{ \begin{array}{lll} \operatorname{argmin}_{x \in X} & \frac{1}{2}\|x - \hat{x}^k\|^2 & \\ \text{s.t.} & (c^{\mathsf{T}} + \alpha_l^{\mathtt{av}})x + \beta_l^{\mathtt{av}} \leq f_{\mathtt{lev}}^k, & l \in J_k^c \\ & (c^{\mathsf{T}} + \alpha_l\,)x + \beta_l\, \leq f_{\mathtt{lev}}^k, & l \in J_k^f \end{array} \right. \tag{19}$$

3.2 Level decomposition with adaptive partitions

We present two level decompositions combined with an adaptive partition scheme. The first and simpler algorithm applies the level bundle method of [20] to solve the partition-based master problem in the adaptive-partition scheme of [32]. Given a partition $\mathcal{P}^\ell$, the resulting recourse mapping $\mathcal{Q}_\ell^{\mathtt{av}}$ is obtained and the algorithm employs the level decomposition to solve the smaller problem (10) and obtain an approximate solution $\hat{x}$. Function $\mathcal{Q}(x)$ is then evaluated at $\hat{x}$ and a partition refinement strategy is applied to refine $\mathcal{P}^\ell$ according to the set of second-stage dual optimal solutions with respect to $\hat{x}$. The process is repeated until a solution to problem (1) is found.

The second algorithm employs the level decomposition and partition scheme in a dynamic manner: instead of solving (10) up to optimality for a given partition $\mathcal{P}^\ell$, the algorithm updates the partition at certain points yielding decrease of the objective function. Moreover, the algorithm has limited memory, does not require boundedness of the feasible set and incorporates the concept of on-demand accuracy.

3.2.1 Level decomposition for solving partition-based master problems.

Algorithm 1 starts with a finite lower bound $f_{\mathtt{low}}^0$ for the optimal value of (1). It could be obtained by solving problem (10) with $L = 1$, $\mathcal{P} = \{N\}$, i.e., the so-called *mean-value problem*. The initial partition could also be chosen by applying either the clustering algorithm (e.g., the K-means algorithm) or scenario reduction techniques. The resulting optimal value is a lower bound by (9).

Algorithm 1 LEVEL DECOMPOSITION WITH ADAPTIVE PARTITIONS

Step 0 (initialization). *Let $k = 0$, $\ell = 0$, $\kappa_f, \kappa \in (0,1)$, and set $J_0^c = J_0^f = \emptyset$. Choose $\hat{x}^0 \in X$, $f_{\texttt{low}}^0$ a given lower bound, a partition $\mathcal{P}^\ell = \{P_1, P_2, \ldots, P_{L_0}\}$, and tolerances $\texttt{Tol}^c \geq \texttt{Tol}^f > 0$. Set $\bar{z}^0 = f_{\texttt{up}} = \infty$.*

Step 1 (coarse stopping test). *If $\bar{z}^k - f_{\texttt{low}}^k \leq \texttt{Tol}^c$, go to Step 6. Otherwise, continue.*

Step 2 (master problem) *Define $f_{\texttt{lev}}^k = \kappa f_{\texttt{low}}^k + (1-\kappa)\bar{z}^k$. If problem* (19) *is feasible, obtain x^{k+1} by solving* (19)*. Otherwise, set $f_{\texttt{low}}^{k+1} = f_{\texttt{lev}}^k$, $\bar{z}^{k+1} = \bar{z}^k$, $J_{k+1}^c = J_k^c$, and $J_{k+1}^f = J_k^f$. Set $\hat{x}^{k+1} = \hat{x}^k$, $k = k+1$ and go back Step 1.*

Step 3 (coarse cut). *Compute $Q(x^{k+1}; \xi_j^{\texttt{av}})$ for all $j = 1, \ldots, L_\ell$, and let $\lambda_j^{\texttt{av}}$ be the associate dual solution. Compute a coarse cut as in* (16)*.*

Step 4 (stability center updating). *Define $\bar{z}^{k+1} = \min\{\bar{z}^k, c^\mathsf{T} x^{k+1} + \mathcal{Q}_\ell^{\texttt{av}}(x^{k+1})\}$. If $\bar{z}^{k+1} < \bar{z}^k - \kappa_f(\bar{z}^k - f_k^{lev})$ set $\hat{x}^{k+1} = x^{k+1}$. Otherwise, $\hat{x}^{k+1} = \hat{x}^k$.*

Step 5 (inner loop). *Update the index sets $J_{k+1}^c = J_k^c \cup \{k+1\}$, $J_{k+1}^f = J_k^f$. Set $f_{\texttt{low}}^{k+1} = f_{\texttt{low}}^k$, $k = k+1$ and go back to Step 1.*

Step 6 (fine stopping test). *If $f_{\texttt{up}} - f_{\texttt{low}}^k \leq \texttt{Tol}^f$, stop: the point $x_{\texttt{best}}$ is an $\texttt{Tol}^f$-solution to problem* (1)*.*

Step 7 (fine cut) *Set $x^{k+1} = \hat{x}^k$, compute $Q(x^{k+1}; \xi_i)$ for all $i \in N$, and let λ_i be the associate dual solution. Compute a fine cut as in* (17)*. If $c^\mathsf{T} x^{k+1} + \mathcal{Q}(x^{k+1}) < f_{\texttt{up}}$, then set $f_{\texttt{up}} = c^\mathsf{T} x^{k+1} + \mathcal{Q}(x^{k+1})$ and $x_{\texttt{best}} = x^{k+1}$.*

Step 8 (new partition). *Obtain a new partition $\mathcal{P}^{\ell+1}$ by using the absolute rule of [32], the K-means rule* (14)*, etc. Set $\bar{z}^{k+1} = f_{\texttt{up}}$, $f_{\texttt{low}}^{k+1} = f_{\texttt{low}}^k$, $\hat{x}^{k+1} = x_{\texttt{best}}$, $J_{k+1}^f = J_k^f \cup \{k+1\}$ and $J_{k+1}^c = J_k^c$. Update $\ell = \ell + 1$, $k = k+1$ and go to Step 1.*

To identify if the level set is empty in Step 2, the most natural way is probably to proceed as usual with solving (19) and let the employed QP solver returns with an infeasibility flag. Note that this is not a wasteful computation, as it leads to adjusting the level parameter as well as improving the lower bound $f_{\texttt{low}}^k$.

At the first few iterations ℓ, the partitions $\mathcal{P}^\ell$ may not represent the whole set of scenarios well. Therefore, it makes sense to inexactly solve the partition-based problem (10). This amounts to index the tolerance $\texttt{Tol}^c$ by ℓ, starting with a larger value for $\texttt{Tol}_0^c$ and decrease $\texttt{Tol}_\ell^c$ along the iterative process.

Step 5 is only accessed after having approximately solved (10) for a given partition $\mathcal{P}^\ell$. When this is the case, the partition $\mathcal{P}^\ell$ is updated to $\mathcal{P}^{\ell+1}$ and the process continues until the second stopping-test at Step 6 is satisfied. Every time a new partition-based problem (10) is defined we employ previously generated (coarse and fine) cuts to warmstart the optimization process without cutting off the solution set of (10). This is possible thanks to inequality (9) and convexity of both functions $\mathcal{Q}$ and $\mathcal{Q}^{\texttt{av}}$, which ensures that any valid cut for $\mathcal{Q}^{\texttt{av}}$ is also valid for function $\mathcal{Q}$, as in (16).

We emphasize that the inner loop consisting of Steps 1-5 is a level bundle method applied to $Q^{\texttt{av}}$, with an exact oracle for the latter. It terminates finitely as a consequence of [20, Theorem 3.5]. Note further that Steps 6-8 are nothing but another level bundle method with exact oracle for $\mathcal{Q}$. Finite convergence is once again ensured by [20, Theorem 3.5]. We formalize this analysis with the following proposition.

Proposition 3.1 *Assume that the set X is compact and that X lies in the interior of the domain of Q. If $\texttt{Tol}^f$ and $\texttt{Tol}^c$ are strictly positive, then Algorithm 1 stops after a finite number of steps with a $\texttt{Tol}^f$-solution $x_{\texttt{best}}$ to problem* (1)*.*

Proof. Note that $\delta' > \delta > 0$ can be found such that $X \subseteq \mathbb{B}^0(0,\delta) \subseteq \mathbb{B}(0,\delta') \subseteq \operatorname{int} \mathcal{D}\mathrm{om}(Q)$ and that convex mappings are Locally Lipschitz on the interior of their domain by [4, Proposition 2.2.6], where $\mathbb{B}(0,\delta')$ is compact. Consequently Q is Lipschitz continuous on any compact set contained in their domain by [25, Theorem 1.14]), hence in particular on $\mathbb{B}^0(0,\delta)$. Say with Lipschitz constant $L > 0$. This implies by [4, Proposition 2.1.2] that any subgradient of Q is bounded in norm by L. Note too that $\mathcal{D}\mathrm{om}(Q^{\texttt{av}}) = \mathcal{D}\mathrm{om}(Q)$ and a similar analysis can be carried out for $Q^{\texttt{av}}$. As a consequence, the inner loop consisting of steps 1-5 is a level bundle method applied to mapping $Q^{\texttt{av}}$, with an exact oracle for the latter. It terminates finitely as a consequence of [20, Theorem 3.5].

Note further that the iterations k during which the inner loop is executed have a fixed iteration counter ℓ. Let $K(\ell)$ regroup this set of indexes. There exists $k(\ell) \in K(\ell)$ such that $\hat{x}^k$ in Step 4, given to the fine oracle in Step 7 on termination of the inner loop is equal to $x^{k(\ell)+1}$ resulting from the solution of problem (19). From this viewpoint steps 6-8 are nothing but another level bundle method with exact oracle. Finite convergence is once again ensured by [20, Theorem 3.5]. □

3.2.2 Level decomposition with a unified oracle using adaptive partitions

In this section, we unify the fine and coarse oracles described in Section 3.1, under the framework of level bundle method with on-demand accuracy. The idea is to maintain a "partly inexact" oracle, a concept introduced by [6], which gives exact function and subgradient information whenever the estimate achieves a certain descent target γ^k. This "partly inexact" oracle is defined by a given partition $\mathcal{P}$, which gives $\alpha^{\mathtt{av}}$ and $\beta^{\mathtt{av}}$ if the descent target γ^k is proved not achievable, and gives exact function and subgradient information by exactly solving all $|N|$ scenario subproblems, otherwise. The second-stage dual solutions will in turn guide the refinement of the partition $\mathcal{P}$. This unified oracle could also be designed as a "partly asymptotically exact" oracle, if the scenario subproblems are solved inexactly given an inexactness threshold ϵ^k, and $\epsilon^k \to 0$. To be consistent with previous sections, we consider the single-cut variant. A multi-cut version can be derived similarly.

Algorithm 2 LEVEL DECOMPOSITION WITH A UNIFIED ORACLE BY ADAPTIVE PARTITIONS.

Step 0 (initialization). *Let $k = 0$, $\ell = 0$, $\kappa, \kappa_f \in (0,1)$, $\tau > 0$, and tolerances $\mathtt{Tol}, \mathtt{Tol}_g > 0$. Choose $x^0 = \hat{x}^0 \in X$ and obtain an initial upper bound $f^0_{\mathtt{up}}$ by computing $Q(\hat{x}; \xi_i)$, $\forall i \in N$. Choose a partition $\mathcal{P}^0 = \{P_1, P_2, \ldots, P_{L_0}\}$ and a lower bound $f^0_{\mathtt{low}}$. Set $v^0 = (1-\kappa)(f^0_{\mathtt{up}} - f^0_{\mathtt{low}})$, $J^c_0 = \emptyset$, and $J^f_0 = \{0\}$.*

Step 1 (first stopping test). *If $f^k_{\mathtt{up}} - f^k_{\mathtt{low}} \le \mathtt{Tol}$, stop: $\hat{x}^k$ is an $\mathtt{Tol}$-solution to problem* (1).

Step 2 (master problem). *Define $f^k_{\mathtt{lev}} = f^k_{\mathtt{up}} - v^k$. If problem* (19) *is feasible, then obtain x^{k+1} by solving* (19) *and corresponding multipliers t_l, with $l \in J^c_k \cup J^f_k$. Define $\tau_k = \sum_{l \in J^c_k \cup J^f_k} t_l$, $\hat{g}^k = (\hat{x} - x^{k+1})/\tau_k$ and $\hat{e}^k = v^k - \tau_k \|\hat{g}^k\|^2$.*

If problem (19) *is infeasible, the update $f^{k+1}_{\mathtt{low}} = f^k_{\mathtt{lev}}$, $f^{k+1}_{\mathtt{up}} = f^k_{\mathtt{up}}$, and $v^{k+1} = (1-\kappa)(f^{k+1}_{\mathtt{up}} - f^{k+1}_{\mathtt{low}})$. Set $J^c_{k+1} = J^c_k$, $J^f_{k+1} = J^f_k$, $k = k+1$ and go back Step 1.*

Step 3 (second stopping test). *If $\hat{e}^k \le \mathtt{Tol}$ and $\|\hat{g}^k\| \le \mathtt{Tol}_g$, stop: the point $\hat{x}$ is an approximate solution to problem* (1).

Step 4 (multiplier attenuation). *If $\tau_k > \tau$, update $v^{k+1} = v^k/2$. Set $f^{k+1}_{\mathtt{low}} = f^k_{\mathtt{low}}$, $f^{k+1}_{\mathtt{up}} = f^k_{\mathtt{up}}$, $J^c_{k+1} = J^c_k$, and $J^f_{k+1} = J^f_k$, $k = k+1$ and go back Step 2.*

Step 5 (oracle call). *Select a new descent target $\gamma^k = f^k_{\mathtt{up}} - \kappa_f v^k$. Compute $Q(x^{k+1}; \xi^{\mathtt{av}}_j)$ for all $j = 1, \ldots, L_\ell$, and let $\lambda^{\mathtt{av}}_j$ be the associate dual solution. Set $f = c^\mathsf{T} x^{k+1} + \mathcal{Q}^{\mathtt{av}}_\ell(x^{k+1})$. If $f > \gamma^k$, then compute a coarse cut as in* (16) *and go to Step 7.*

Step 6 (partition refinement). *Choose a cluster $j \in \{1, \ldots, L_\ell\}$, and do the following tasks:*

Compute $Q(x^{k+1}; \xi_i)$ for all $i \in \mathcal{P}^\ell_j$ and let λ_i be the associate dual solution. Refine component $\mathcal{P}^\ell_j$ guided by dual solutions $\lambda_i, i \in \mathcal{P}^\ell_j$ using a refinement strategy. Improve the estimate function value by $f = f + (\sum_{i \in \mathcal{P}^\ell_j} p_i Q(x^{k+1}) - \pi_j Q(x^{k+1}; \xi^{\mathtt{av}}_i)$. If $f > \gamma^k$, compute a semi-coarse cut (16) *using mixed coarse/fine information: use λ_i if available, otherwise use $\lambda^{\mathtt{av}}_j$. Terminates this Step and move to Step 7. If $f \le \gamma^k$, choose a different $j \in \{j = 1, \ldots, L_\ell\}$ and repeat this process. When this Step is finalized, set $\ell = \ell + 1$.*

Step 7 (stability center updating). *If $f < \gamma^k$, declare a Serious Step. Update $f^{k+1}_{\mathtt{up}} = f$, $\hat{x}^{k+1} = x^{k+1}$, $v^{k+1} = \min\{v^k, f^{k+1}_{\mathtt{up}} - f^k_{\mathtt{low}}\}$. Choose J^f_{k+1} such that $k+1 \in J^f_{k+1}$ and $J^c_{k+1} \subset J^c_k$ (e.g. $J^f_{k+1} = \{k+1\}$ and $J^c_{k+1} = \emptyset$). If $f \ge \gamma^k$, declare a Null Step. Set $f^{k+1}_{\mathtt{up}} = f^k_{\mathtt{up}}$, $\hat{x}^{k+1} = \hat{x}^k$ and $v^{k+1} = v^k$. Choose J^f_{k+1} such that $\{l \in J^f_k : t_l > 0\} \subset J^f_{k+1}$ and choose J^c_{k+1} such that $\{\{l \in J^c_k : t_l > 0\} \cup \{k+1\}\} \subset J^c_{k+1}$.*
In any case, set $f^{k+1}_{\mathtt{low}} = f^k_{\mathtt{low}}$, $k = k+1$ and go to Step 1.

Algorithm 2 is a particular version of [6, Algorithm 4.2], corresponding to variant PI2'. As a result, the convergence analysis of Algorithm 2 follows from [6, Theorem 4.7]. The following proposition gives a formalization of this assertion.

Proposition 3.2 *Assume that the set X lies in the interior of the domain of Q. Suppose that $\mathtt{Tol} = \mathtt{Tol}_g = 0$. Then the sequence $\{\hat{x}^k\}$ converges to a point that solves* (1). *Moreover, at least one the following items hold true i) $\lim_k f^k_{\mathtt{up}} - f^k_{\mathtt{low}} = 0$, ii) $\liminf_k \max\{\hat{e}^k, \|\hat{g}^k\|\} = 0$, where $\hat{e}^k$ and $\hat{g}^k$ are defined in Step 2 of Algorithm 2.*

Proof. Note that function $\mathcal{Q}$ and a subgradient is exactly computed at every serious step. Therefore, Algorithm 2 corresponds to variant PI2' of Algorithm 4.2 in [6]. Regardless of the accuracy of the computed subgradient for $\mathcal{Q}(x^k)$ (coarse $\alpha^{\mathtt{av}}_k$ or fine α_k), there exists a constant $\Lambda > 0$ (c.f. the proof of Proposition 3.1) such that $\max\{\|c + \alpha^{\mathtt{av}}_k\|, \|c + \alpha_k\|\} \le \Lambda$. Thus Lemma 4.3 of [6] yields the following inequality:

$$\|x^{k+1} - x^k\| \ge (1 - \kappa_f)\frac{v^k}{\Lambda}.$$

Since x^{k+1} is the projection of $\hat{x}^k$ onto a convex set, combined with the above inequality, we have that

$$\|x^{k+1} - \hat{x}\|^2 \geq \|x^k - \hat{x}\|^2 + \left((1-\kappa_f)\frac{v^k}{\Lambda}\right)^2,$$

for all iterations issued by the same stability center $\hat{x}^k = \hat{x}$. By using this inequality, Propositions 4.5 and 4.6 of [6] ensure that the sequence $\{v^k\}$ vanishes regardless of whether $X \subseteq \mathbb{R}^n$ is bounded or not. Since the sequence $\{v^k\}$ vanishes, Lemma 4.4 of [6] ensures that i) or ii) holds. Convergence of the whole sequence $\{\hat{x}^k\}$ is therefore ensured by [6, Propositions 4.5]; see also [6, Theorem 4.7]. □

4 Numerical results

We conduct numerical experiments to test the empirical performances of the proposed approaches. We implement all algorithms in C++ using the commercial solver CPLEX, version 12.5.1. All tests are conducted on a Linux workstation with four 3.00GHz processors and 8Gb memory. The number of threads is set to be one. We use an open source numerical linear algebra package, Eigen [14] for large-scale matrix and vector operations.

In all our implementation, the starting solution is given by solving the mean-value problem. We use a convergence criterion implying that the relative optimality gap $(UB - LB)/UB < 10^{-4}$, where UB and LB are the best upper and lower bound obtained by the algorithm, respectively. After having conducted several tests to tune parameters, we take the best ones and set the second stopping tolerance parameter $\texttt{Tol}_g = \sqrt{n_1} \times 10^{-4}$, and let $\tau = 10$ in Algorithm 2. In our implementation of the cutting plane method (both the single-cut version and the multi-cut version), we solve the master problem (10) using the dual simplex method by CPLEX. We add a cut $\theta \geq \alpha x + \beta$ when the current relaxation solution $(\hat{\theta}, \hat{x})$ violates the cut by more than a violation threshold of $\max\{1, |\hat{\theta}|\} \times 10^{-4}$. For Algorithm 1 we use parameters $\kappa = 0.8$, $\kappa_f = 0.1$, and we set the convergence threshold for the partition-based master problem (10) also using the relative optimality gap. For Algorithm 2, we use parameters $\kappa = 0.8$ and $\kappa_f = 0.3$.

We report the average results over five replications for each instance and sample size. We use the following abbreviations in all tables throughout this section: Time (solution time in seconds), Size (average partition size), and Iter (number of iterations). For all our tests, we use a time limit of 10800 seconds (three hours). When the time limit is exceeded by any of the five replications, we report instead the average optimality gap obtained when the time limit is met for those instances that cannot be solved within the time limit, and show in parenthesis the number of replications solved to optimality. If all five replications cannot be solved within the time limit, we just report the average optimality gap, calculated by $(UB - LB)/UB$.

4.1 Algorithmic benchmark

In our experiments, we test the following variants of the proposed algorithms: Single-`lvl` (Algorithm 1), Single-`lvl-oda` (Algorithm 2), and Single-`cp` (unregularized version of Algorithm 1, i.e., an overall cutting plane approach). We also implemented the multi-cut versions of these variants, which yielded better performances in some instances. For the multi-cut version, since the variable space of the partition-based master problem formulation (10) changes as the partition is refined, a new master problem is created for each partition-based master problem. In this case, we store the cut (dual multiplier) information together with the partition, so that these cuts are reused for the new formulation after a partition refinement. For the single-cut version, we can just keep a single master problem throughout the algorithm, and the cut information can be carried over automatically from the original partition to its refined one.

4.2 Instances

We use instances on two-stage stochastic programs with fixed recourse described in [22] (available at `http://pages.cs.wisc.edu/~swright/stochastic/sampling/`) and [8] as a basis to generate our test cases. The probability distributions of some instances are specified, and in this case, we generate random samples with various sample sizes following these distributions. For example, in instance "gbd",

"ssn", and "LandS", each random variable follows a discrete probability distribution independently. In other instances, the probability distribution is implicitly given as a set of scenarios, or a set of blocks. A scenario represents a realization of all random variables, and a block represents a realization of a subset of random variables. For these instances, for example, the "DEAK" family of instances and "stormG2", we compute the sample mean $\hat{\mu}$ and sample standard deviation $\hat{\sigma}$ of each random variable according to the set of scenarios or blocks, and then generate random samples with various sample sizes following a normal distribution $\mathcal{N}(\hat{\mu}, \frac{2}{3}\hat{\sigma})$. Table 1 provides the sizes of these instances.

Instance	First-stage size	Second-stage size
DEAK20×20	(20,10)	(30,20)
DEAK20×40	(20,10)	(60,40)
DEAK20×60	(20,10)	(90,60)
DEAK40×20	(40,20)	(30,20)
DEAK40×40	(40,20)	(60,40)
DEAK40×60	(40,20)	(90,60)
DEAK60×20	(60,30)	(30,20)
DEAK60×40	(60,30)	(60,40)
DEAK60×60	(60,30)	(90,60)
LandS	(2, 4)	(7, 12)
gbd	(4, 17)	(5, 10)
stormG2	(185,121)	(528,1259)
ssn	(1,89)	(175,706)

Table 1: Profiles of test instances from [22] and [8]. We use (n, m) to denote that the number of variables is n, and the number of constraints is m.

We consider three scenario sizes for each instance family: 1k, 5k and 10k for "stormG2" and "ssn", and 20k, 50k and 100k for all other instances. Although the sample sizes used in our experiments for instances "LandS" and "gbd" may be a little too big for their problem sizes, which can be verified using an in-sample and out-of-sample stability test (see, e.g., [19, Chap.4]), we emphasize that our motivation of using large scenario sizes for these instances is to show the scalability of the proposed approach compared to the existing alternatives, but not to solve the underlying stochastic program with a given continuous probability distribution to a certain accuracy. All the test instances are available at `https://sites.google.com/site/yongjiasongshom/research`.

4.3 Results

Table 2 shows the performances of our solvers Single-`cp`, Single-`lvl`, and Single-`lvl-oda` (with an exception of instance *ssn*, where the multi-cut version yielded a better performance). Notice that the three solvers have comparable performances for easy instances, such as DEAK20x20, DEAK20x40, DEAK40x20 and DEAK40x40. However, solver Single-`cp` is outperformed by Single-`lvl` and Single-`lvl-oda` in larger instances. For instance, Single-`cp` required 5742.7 seconds on average to solve instance DEAK60x40 with one hundred thousand scenarios, whereas the level solvers were approximately 77% faster. Similar results were obtained for other difficult problems. Concerning the number of iterations, we record the number of times that the partition-based master problem is solved in Single-`cp` and Single-`lvl`, and we record the number of times that the fine and coarse oracles are called in Single-`lvl-oda`. We see that Single-`lvl-oda` yielded significantly more iterations switching between the fine and coarse cuts, which indicates the additional flexibility provided by this algorithm. This additional flexibility does not lead to significant saving in computational time, and solvers Single-`lvl` and Single-`lvl-oda` gave comparable computational performances on our test instances. We also observe that solver Single-`lvl-oda` yielded partitions of significantly larger sizes than solvers Single-`cp` and Single-`lvl`. This is due to the fact that more partition refinements are performed in Single-`lvl-oda`.

Table 3 shows the performances of existing algorithms proposed in the literature on the test instances. In solver `Best-Benders`, we choose the better one between the single-cut (L-shaped method) and multi-cut version of Benders decomposition in terms of their average computational performance. In particular,

Instances		Single-cp			Single-lvl			Single-lvl-oda		
Instance	$\|N\|$	Time	Size	Iter	Time	Size	Iter	Time	Size	Iter
DEAK20x20	20k	15.2	232	7	14.5	215	7	18.7	207	17
	50k	38.5	301	7	38.3	292	7	44.3	238	16
	100k	72.5	376	6	83.6	393	7	90.7	280	16
DEAK20x40	20k	15.3	71	3	12.1	53	3	20.3	118	7
	50k	29.0	51	3	30.8	60	3	49.5	113	6
	100k	67.9	84	3	79.8	116	4	90.0	108	6
DEAK20x60	20k	258.5	7696	4	138.5	7484	4	151.5	10238	22
	50k	612.6	15428	4	266.9	13633	4	345.3	22231	21
	100k	1651.6	31534	4	484.2	25742	4	680.1	40076	21
DEAK40x20	20k	16.0	234	6	14.8	212	6	19.7	348	17
	50k	47.7	419	7	41.8	302	7	52.2	480	19
	100k	79.4	416	6	73.1	358	7	104.1	484	17
DEAK40x40	20k	32.6	1282	4	29.2	1369	5	41.3	2645	14
	50k	79.0	2052	4	73.1	2220	5	104.8	4660	13
	100k	173.4	3109	4	125.0	3071	5	199.5	6872	13
DEAK40x60	20k	356.8	7756	4	215.4	7247	4	190.5	12591	23
	50k	916.7	17207	4	571.7	17812	4	480.1	29336	24
	100k	2656.5	37985	5	848.6	28638	4	878.6	53739	23
DEAK60x20	20k	165.4	4030	6	84.5	3587	6	109.9	8396	80
	50k	463.2	7880	6	173.4	6542	6	221.4	16487	75
	100k	1562.5	14342	6	264.7	9759	6	387.3	26093	73
DEAK60x40	20k	1170.1	8470	4	485.8	8496	4	434.8	14997	84
	50k	4540.6	19700	5	965.0	19273	5	1155.7	37015	91
	100k	5742.7	31366	4	1707.9	34377	5	2049.1	68509	86
DEAK60x60	20k	607.5	8537	4	340.7	7628	4	352.0	15172	36
	50k	1668.3	19561	4	741.3	17367	4	745.5	33657	35
	100k	3275.8	34953	4	1312.3	34985	5	1423.6	62206	34
LandS	20k	4.2	106	5	4.7	161	6	4.9	160	15
	50k	10.0	122	5	11.7	168	6	11.0	148	15
	100k	20.8	110	5	23.3	178	6	24.9	179	15
gbd	20k	6.6	427	5	6.4	460	5	10.5	614	29
	50k	16.6	564	6	14.0	492	5	24.0	617	27
	100k	27.5	453	5	27.5	496	5	51.4	989	29
stormG2	1k	130.5	369	3	56.8	393	3	63.7	533	23
	5k	453.4	1409	3	182.3	1485	3	249.1	2187	27
	10k	830.6	2308	3	303.5	2712	3	389.8	3796	26
ssn†	1k	65.2	567	3	176.2	606	4	303.7	815	17
	5k	335.0	2613	3	1044.3	2682	4	377.7	3845	17
	10k	759.5	5002	4	1445.6	5006	4	827.8	7306	19

'†': A multicut version is implemented for this family of instances.

Table 2: Average time, average partition size, and number of iterations for solvers Single-cp, Single-lvl, and Single-lvl-oda for solving two-stage stochastic programs with fixed recourse.

in both implementations of Benders decomposition, we solve the Benders master problem using the barrier method by CPLEX. We add a Benders cut when the relaxation solution violates the cut by more than a violation threshold. We set this threshold to be $\max\{1, |Q(\hat{x}, \xi_i)|\} \times 10^{-4}$ for the multi-cut version for each scenario $i \in N$, and $\max\{1, |\sum_{i \in N} p_i Q(\hat{x}, \xi_i)|\} \times 10^{-4}$ for the single-cut version, where $\hat{x}$ is the current relaxation solution. In solver `Level`, we implement the level method, which can be seen as applying Algorithm 1 to a partition $\mathcal{P} = \{\{1\}, \{2\}, \ldots, \{|N|\}\}$, which is essentially the original stochastic program (1). We emphasize that solvers `Best-Benders` and `Level` do not employ any partition scheme. In solver `Merge-Partial`, we pick the best computational strategy for the adaptive partition-based approach proposed in [32]. We use the bold font to indicate that the result corresponds to the multi-cut implementation.

Instances		`Best-Benders`		`Level`		`Merge-Partial`		
Instance	$\|N\|$	Time	Iter	Time	Iter	Time	Size	Iter
DEAK20x20	20k	37.8	16	31.8	40	15.0	67	7
	50k	94.1	16	73.8	38	37.5	80	7
	100k	191.2	16	142.6	37	76.5	94	7
DEAK20x40	20k	**58.4**	**7**	28.2	22	15.5	38	3
	50k	**153.8**	**7**	71.9	22	39.2	45	3
	100k	**323.5**	**7**	141.7	22	77.0	48	3
DEAK20x60	20k	**130.6**	**8**	210.2	39	4697.6	4200	4
	50k	**371.0**	**8**	524.1	38	0.2%(0)	7370	>3
	100k	**890.6**	**8**	1093.2	39	0.2%(0)	11545	>3
DEAK40x20	20k	55.5	20	35.0	39	14.1	74	6
	50k	138.4	20	90.1	39	34.2	90	6
	100k	272.4	19	189.7	39	69.5	99	6
DEAK40x40	20k	150.7	21	86.4	40	43.7	463	5
	50k	396.0	22	231.5	41	94.2	646	5
	100k	810.8	22	471.6	41	171.4	806	5
DEAK40x60	20k	**333.5**	**13**	332.4	48	5724.6	4658	4
	50k	**923.7**	**13**	858.2	49	0.4%(0)	8301	>3
	100k	**1934.4**	**13**	1624.6	48	0.4%(0)	13158	>3
DEAK60x20	20k	**297.9**	**13**	182.0	96	28.4	1056	6
	50k	**840.7**	**13**	453.6	96	61.5	1679	6
	100k	**1914.7**	**13**	877.6	94	107.6	2080	6
DEAK60x40	20k	**604.0**	**17**	611.8	110	893.5	4197	5
	50k	**1669.1**	**17**	1432.3	106	4463.8	7562	5
	100k	**3668.3**	**17**	2962.8	109	0.1%(2)	10794	>5
DEAK60x60	20k	**477.0**	**13**	436.7	57	4788.0	4990	3
	50k	**1334.8**	**13**	1073.0	57	0.5%(0)	9481	>3
	100k	**2858.4**	**13**	2174.3	57	0.5%(0)	15031	>3
LandS	20k	20.9	19	15.2	34	4.7	41	5
	50k	54.8	20	36.6	34	11.6	42	5
	100k	110.8	20	69.7	33	23.5	41	5
gbd	20k	32.2	27	25.3	42	5.0	135	5
	50k	79.5	27	59.4	41	12.3	142	5
	100k	164.5	27	119.5	42	24.7	143	5
stormG2	1k	276.4	84	126.4	49	72.9	336	3
	5k	1350.8	85	570.7	46	379.3	1163	3
	10k	2838.8	87	1131.5	45	764.7	1915	3
ssn	1k	**99.3**	**17**	145.8	45	110.1	381	5
	5k	**478.5**	**14**	619.5	38	5763.5	1501	7
	10k	**932.3**	**13**	1395.1	36	-†	-	-

'†': Cannot solve the partition-based master problem after the first refinement.

Table 3: Computational performances of the Benders decomposition (the better one between the single-cut and multi-cut version), the level method, and the `Merge-Partial`solver of [32]. `Best-Benders`and `Level`solvers do not employ the partition-based scheme.

We can see from Table 3 that our proposed approaches work significantly better than the one proposed in [32] in term of overall computational time. Indeed, [32] solve the partition-based master problem (10) to optimality using the extended formulation, which could be time consuming to solve if the partition size is large. Comparing solver `Merge-Partial` in Table 3 and solvers Single-`cp` and Single-`lvl` in Table 2, we can see that in the setting where a partition-based master problem (10) is solved to optimality before the par-

tition is refined, it is computational advantageous to solve the master problem using a cutting plane method or level method, rather than solving it as a deterministic equivalent linear program. The proposed Algorithm 2 allows the partition to be refined before the partition-based master problem is solved to optimality, which brings in additional flexibility. We next compare solvers `Best-Benders` and `Level` in Table 3 with solvers Single-`cp` and Single-`lvl` in Table 2, respectively. Benders decomposition and level bundle method are applied to the scenario-based formulation in `Best-Benders` and `Level`, and on the other hand, they are applied in Single-`cp` and Single-`lvl` to solve the adaptively refined partition-based formulation. In general, we can clearly see that it is computational advantageous to apply the adaptive partition-based framework on the test instances. However, for several instances, solver `Best-Benders` performed better than its counterpart Single-`cp`, which indicates that the adaptive partition scheme is not always the best alternative when dealing with a pure cutting-plane model. On the other hand, when employed with the level decomposition (Single-`lvl` or Single-`lvl-oda`), the partition-based scheme provided a better performance than the (pure) level decomposition in all but one instance. Within the level decomposition, the partition-based approach provided CPU time reduction of the order of 52% on average.

In order to analyse our results and put them in a convenient form, we next use the performance profiles [9]. For each solver, we plot the proportion of problems that are solved within a factor of the time required by the best algorithm: if we denote by $t_S(p)$ the time spent by solver S to solve problem p and $t^*(p)$ the best time for solving problem p, then the proportion of problems solved by S within a factor γ is

$$\rho_S(\gamma) = \frac{\text{number of problems } p \text{ such that } t_S(p) \leq \gamma\, t^*(p)}{\text{total number of problems}}.$$

The leftmost ordinate value gives the probability of each solver to be the fastest in the benchmark, while robustness is seen by the rightmost ordinate value, with the proportion of problems solved by each solver.

Figure 4.3 exhibits the performance profiles based on CPU times of different solvers. It can be seen that the Single-`lvl` solver, closely followed by Single-`lvl-oda`, outperforms other solvers, both in terms of robustness as well as speed. In particular, although the `Merge-Partial` solver was the fastest solver in around 23% of all considered problems, eventually it is outperformed by `Best-Benders`. The level decomposition is also advantageous over `Best-Benders`, and becomes more efficient when combined with the adaptive partition strategy. In particular, Single-`lvl` can solve around 85% of the instances nearly four to five times faster than the level decomposition alone, and obtain a speedup of seven times faster than `Best-Benders`. We believe that these performance profiles exhibit the potential of combining both features.

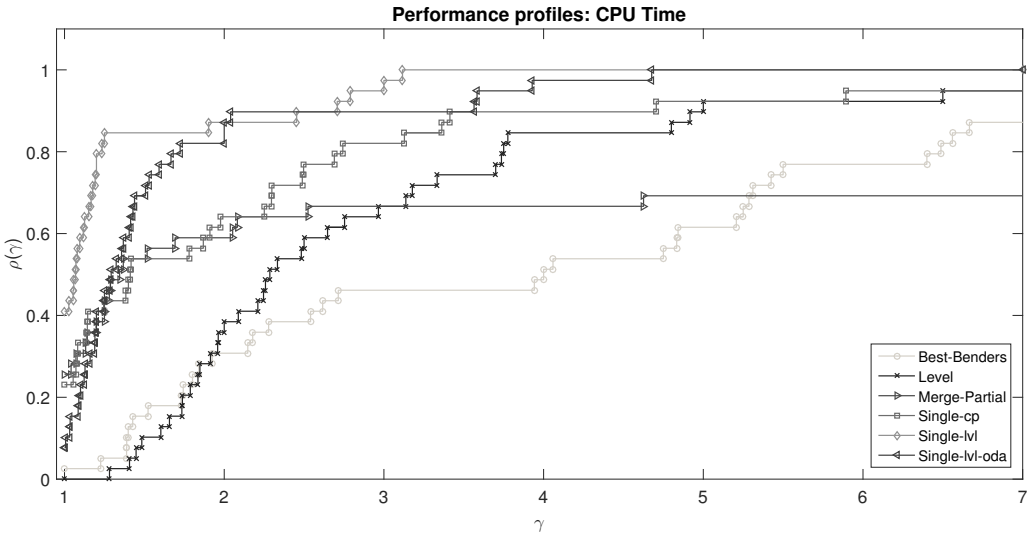

Figure 1: Performance profiles on CPU time for different solvers presented in Tables 2 and 3.

4.4 Computational performances for various refinement strategies

We next show the computational performances of the partition-based level decomposition using various refinement strategies. To perform the K-means clustering, we use the C-Clustering library[1]. In our computational experiments, we observe that both the K-means clustering algorithm and the fast forward heuristic algorithm for the scenario reduction [15] are much more time-consuming than the "Absolute" strategy by [32]. Therefore, we design the refinement strategies in a hybrid manner so that the K-means clustering or the scenario reduction algorithm is performed only if the absolute strategy fails to yield a small number of new partition components after a refinement. Specifically, we compare the computational performances of the following strategies:

- Absolute: Only apply the "absolute" refinement strategy (13) by putting similar scenarios together according to their pairwise distances.
- Cluster: Suppose a partition component S is to be refined. If $|S| \leq 20$, we just perform the "absolute" refinement strategy according to (13). Otherwise, we first apply the "absolute" refinement strategy on this partition component S, and suppose we obtain S' new components after the refinement. We then use the following strategy to control the size of the new partition after the refinement: if $S' < \frac{|S|}{5}$, we just include these new components in the new partition; otherwise, we cluster scenarios in S using the K-means clustering algorithm [24] with $K = \min\{10, |S|/10\}$.
- Reduction: The same strategy is used as "Cluster", except that a scenario reduction problem (15) is solved instead of the clustering algorithm. We set $L = |S| - \min\{10, |S|/10\}$ in (15), so that $\min\{10, |S|/10\}$ representative scenarios are selected in S and other scenarios are assigned to these representative scenarios according to their distances. A fast forward algorithm [15] is used to obtain a quick heuristic solution of (15).

We next show the computational performances of the above three refinement strategies using solver Single-`lvl-oda`. We noticed that the performances of the three strategies are very similar when the "absolute" refinement strategy yields a small partition. This is the case for instances DEAK20x20, DEAK20x40, DEAK40x20, DEAK40x40, LandS, and gbd. We show the performances on the rest of the instances in Table 4.

We can see from Table 4 that in most instance, strategies "Absolute" and "Cluster" yielded comparable computational performances. Strategy "Cluster" yielded slightly smaller partitions, but on the other hand took slightly more iterations in most cases. Strategy "Reduction" yielded a more significant partition size reduction, however, at a price of much more iterations, which resulted in more computational time in most cases. In particular, strategy "Reduction" failed to solve instance ssn with 10k scenarios within the time limit. Clearly there is a trade-off between the partition size and the number of iterations using different strategies of partition refinements.

5 Conclusions

This work extends [32] in both theoretical and computational aspects. Concerning the theoretical side, we show that for two-stage stochastic linear programs with fixed recourse, there exists a sufficient partition whose size is independent of number of scenarios, which extends the result in [32] on the special case of simple recourse. On the computational side, we show how level decomposition can be integrated with the adaptive partition-based framework to efficiently solve two-stage stochastic programs with a large scenario set. Two level bundle algorithms based on this idea have been developed. Numerical experiments have shown that both adaptive partition-based level decomposition methods perform significantly better than the cutting-plane methods such as the L-shaped method with or without the partition-based framework, the level method without the partition-based framework, and the extended formulation by [32]. Although the two partition-based level decomposition approaches gave comparable numerical performance, we emphasize that Algorithm 2 is a more general algorithm, and provides more flexibility switching between fine

[1] `http://bonsai.hgc.jp/~mdehoon/software/cluster/software.htm`

Instances		Absolute			Cluster			Reduction		
Instance	$\|N\|$	Time	Size	Iter	Time	Size	Iter	Time	Size	Iter
DEAK20x60	20k	151.5	10238	21	122.0	4600	22	213.5	6935	24
	50k	345.3	22231	20	292.2	18953	18	333.9	18223	21
	100k	680.1	40076	20	570.5	33674	19	693.5	32695	23
DEAK40x60	20k	190.5	12591	23	186.4	11475	23	263.1	11081	33
	50k	480.1	29336	24	490.5	25728	26	577.9	22615	32
	100k	878.6	53739	23	923.4	48777	26	1276.0	43165	37
DEAK60x20	20k	109.9	8396	79	106.3	8003	78	120.0	8230	85
	50k	221.4	16487	75	213.4	15808	73	264.4	15998	86
	100k	387.3	26092	73	415.5	27238	76	532.0	30509	87
DEAK60x40	20k	434.8	14996	84	446.8	14383	88	561.2	14269	112
	50k	1155.7	37015	91	1088.3	35249	88	1481.8	34274	122
	100k	2049.1	68508	86	2055.3	64779	90	2264.2	60656	105
DEAK60x60	20k	352.0	15172	36	307.0	11667	35	499.6	12810	49
	50k	745.4	33657	34	724.7	29440	36	1078.7	30735	52
	100k	1423.6	62206	34	1431.1	56714	37	1969.9	55882	51
stormG2	1k	63.7	533	23	34.1	273	26	31.2	205	24
	5k	249.1	2187	26	169.5	1340	23	175.9	1436	25
	10k	389.8	3796	26	372.3	2948	28	402.6	3189	30
ssn	1k	303.7	814	17	333.0	723	19	734.8	528	119
	5k	377.7	3845	16	429.5	3479	20	9479.0	2219	469
	10k	827.8	7306	18	929.6	6584	21	80.0%(0)	>887	>198

Table 4: Average time, average partition size, and number of iterations for the partition-based level decomposition (Algorithm 2) with different refinement strategies for solving two-stage stochastic programs with fixed recourse.

and coarse oracles. We also note that the proposed approaches are readily applicable to two-stage stochastic integer programs with integer variables only in the first stage.

We have identified several future research directions. First, the proposed approaches could be further improved through specialized acceleration strategies developed for Benders-type algorithms, see. e.g., a recent survey by [28] on Benders decomposition. Second, although the proposed approaches can handle instances with very large number of scenarios, instances used in our test remain small with respect to the number of variables involved in both stages. Analyzing how the proposed approaches would behave when applied on instances with a large number of decision variables, especially the ones with a combinatorial nature, would be an interesting follow-up study to conduct. Finally, the partition-based scheme can be extended for multistage stochastic linear programming problems in, at least, two manners: (a) applying the partition-based framework in the *nested decomposition* [3], and (b) extending the partition-based level decomposition for solving dual formulations arising from the relaxation of the *nonanticipative constraints*, [31, Chap.3].

ACKNOWLEDGMENT Yongjia Song acknowledges partial support by the National Science Foundation under grant CMMI 1562245. 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. The authors greatly appreciate valuable comments and suggestions from the editors and two anonymous referees.

References

[1] B. Bank, J. Guddat, D. Klatte, B. Kummer, and K. Tammer. *Non-Linear Parametric Optimization*. Birkhäuser Basel, 1982.

[2] J. R. Birge. Aggregation bounds in stochastic linear programming. *Mathematical Programming*, 31(1):25–41, May 1985.

[3] J.R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer, New York, 1997.

[4] F.H. Clarke. *Optimisation and Nonsmooth Analysis*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1987.

[5] T.G. Crainic, M. Hewitt, and W. Rei. Scenario grouping in a progressive heding-based meta-heuristic for stochastic network design. *Computers and Operations Research*, 43:90–99, 2014.

[6] W. de Oliveira and C. Sagastizábal. Level bundle methods for oracles with on demand accuracy. *Optimization Methods and Software*, 29(6):1180–1209, 2014.

[7] W. de Oliveira, C.A. Sagastizábal, and S. Scheimberg. Inexact bundle methods for two-stage stochastic programming. *SIAM Journal on Optimization*, 21(2):517–544, 2011.

[8] I. Deák. Testing successive regression approximations by large-scale two-stage problems. *Annals of Operations Research*, 186:83–99, 2011.

[9] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[10] J. Dupačová, N. Gröwe-Kuska, and W. Römisch. Scenario reduction in stochastic programming : An approach using probability metrics. *Mathematical Programming*, 95(3):493–511, March 2003.

[11] D. Espinoza and E. Moreno. A primal-dual aggregation algorithm for minimizing conditional-value-at-risk in linear programs. *Computational Optimization and Applications*, 59:617–638, 2014.

[12] C.I. Fábián. Bundle-type methods for inexact data. In *Proceedings of the XXIV Hungarian Operations Researc Conference (Veszprém, 1999)*, volume 8 (special issue, T. Csendes and T. Rapcsk, eds.), pages 35–55, 2000.

[13] C.I. Fábián, C. Wolf, A. Koberstein, and L. Suhl. Risk-averse optimization in two-stage stochastic models: computational aspects and a study. *SIAM Journal on Optimization*, 25(1):28–52, 2015.

[14] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[15] H. Heitsch and W. Römisch. Scenario reduction algorithms in stochastic programming. *Computation Optimization and Applications*, 24(2-3):187–206, 2003.

[16] H. Heitsch and W. Römisch. A note on scenario reduction for two-stage stochastic programs. *Operations Research Letters*, 35(6):731–738, 2007.

[17] J. L. Higle and S. Sen. Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics Of Operations Research*, 16(3):650–669, 1991.

[18] D.L.D.D. Jardim, M.E.P. Maceira, and D.M. Falcao. Stochastic streamflow model for hydroelectric systems using clustering techniques. In *Power Tech Proceedings, 2001 IEEE Porto*, volume 3, page 6 pp. vol.3, 2001.

[19] A. King and S. Wallace. *Modeling with Stochastic Programming*. Springer, New York, 2012.

[20] K.C. Kiwiel. Proximal level bundle methods for convex nondifferentiable optimization, saddle-point problems and variational inequalities. *Math. Programming*, 69(1):89–109, 1995.

[21] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Math. Programming*, 69(1):111–147, 1995.

[22] J. Linderoth, A. Shapiro, and S. Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142:215–241, 2006.

[23] I. Litvinchev and V. Tsurkov. *Aggregation in Large-Scale Optimization*. Applied Optimization 83. Springer US, 1 edition, 2003.

[24] J. MacQueen. Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967)., 1967.

[25] N. G. Markley. *Principles of Differential Equations*, volume 352 of *Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts*. Wiley-interscience, 1st edition, 2004.

[26] J.M. Morales, S. Pineda, A.J. Conejo, and M. Carrión. Scenario reduction for futures market trading in electricity markets. *IEEE Transactions on Power Systems*, 24(2):878–888, May 2009.

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

[28] R. Rahmaniani, T.G. Crainic, M. Gendreau, and W. Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2017.

[29] C. H. Rosa and S. Takriti. Improving aggregation bounds for two-stage stochastic programs. *Operations Research Letters*, 24(3):127–137, 1999.

[30] B. Sandikçi, N. Kong, and A. J. Schaefer. A hierarchy of bounds for stochastic mixed-integer programming. *Mathematical Programming*, 138(1):253–272, 2013.

[31] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming. Modeling and Theory*, volume 9 of *MPS-SIAM series on optimization*. SIAM and MPS, Philadelphia, 2009.

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

[33] W. van Ackooij. A comparison of four approaches from stochastic programming for large-scale unit-commitment. *To Appear in EURO Journal on Computational Optimization*, pages 1–19, 2015.

[34] W. van Ackooij and W. de Oliveira. Level bundle methods for constrained convex optimization with various oracles. *Computation Optimization and Applications*, 57(3):555–597, 2014.

[35] W. van Ackooij and J. Malick. Decomposition algorithm for large-scale two-stage unit-commitment. *Annals of Operations Research*, 238(1):587–613, 2016.

[36] R.M. van Slyke and R.J-B. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal of Applied Mathematics*, 17:638–663, 1969.

[37] C. Wolf, C. I. Fábián, A. Koberstein, and L. Stuhl. Applying oracles of on-demand accuracy in two-stage stochastic programming a computational study. *European Journal of Operational Research*, 239(2):437–448, 2014.

[38] S. E. Wright. Primal-dual aggregation and disaggregation for stochastic linear programs. *Mathematics of Operations Research*, 19(4):893–908, 1994.