Active Set-based Inexact Proximal Bundle Algorithm for Stochastic Quadratic Programming

Niloofar Fadavi*[1] and Harsha Gangammanavar†[1]

[1]Department of Operations Research and Engineering Management, Southern Methodist University, Dallas TX

Abstract

In this paper, we examine two-stage stochastic quadratic programming problems, where the objective function of the first and second stages are quadratic functions, and the constraints are linear. The uncertainty is associated with the second-stage right-hand side and variable bounds. In large-scale settings, when the number of scenarios necessary to represent the underlying stochastic process is exuberantly large, standard decomposition-based methods that require the exact solutions are computationally prohibitive. To deal with this issue, we develop two inexact proximal bundle algorithms that rely on efficient reutilization of solution information. The first algorithm utilizes a collection of previously encountered second-stage dual solutions to construct inexact minorants for the expectation-valued objective function. On the other hand, a partition-based inexact proximal bundle algorithm utilizes the optimal active sets obtained in earlier iterations and a primal-dual active set method in constructing the inexact minorant. For both these variants, we establish their asymptotic convergence to optimal solutions. Using a carefully developed computer implementation, we demonstrate the practical behavior of these algorithms through numerical experiments conducted on power systems planning and operations problems. The results indicate that the partition-based algorithm consistently identifies solutions that are comparable in quality to those obtained from exact algorithms while significantly reducing the computational time.

History: First submission: Jan 2024; Current version February 2, 2024.

*nfadavi@smu.edu

†harsha@smu.edu

1 Introduction

In this paper, we study the two-stage stochastic quadratic programming (2-SQP) problem that is stated as follows:

$$\begin{aligned} \min\ & f(x) := \frac{1}{2}x^\top Px + c^\top x + \mathbb{E}[h(x,\tilde{\omega})] \\ \text{subject to } & x \in \mathcal{X} = \{x : Ax = b,\ \ell_x \le x \le u_x\}. \end{aligned} \tag{1a}$$

Here, $h(\cdot)$ is the optimal value of the following second-stage quadratic programming (QP) problem:

$$\begin{aligned} h(x,\omega) := \min\ & \frac{1}{2}y^\top Qy + d^\top y \\ \text{subject to } & Dy = \xi(\omega) - C(\omega)x,\ \underline{y}(\omega) \le y \le \bar{y}(\omega),\ y \in \mathbb{R}^{n_2}. \end{aligned} \tag{1b}$$

In the two-stage setting, the first-stage decision $x \in \mathbb{R}^{n_1}$ is determined before the realization of uncertainty that is represented by the random vector $\tilde{\omega}$. The first-stage objective is to minimize the sum of a quadratic function of x and the expected second-stage value function. The expectation is computed with respect to the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ associated with random vector $\tilde{\omega}$. Here, Ω is the set of sample space, $\mathcal{F}$ is the sigma-algebra, and $\mathbb{P}$ is the probability measure defined on $\mathcal{F}$. In the second stage, a QP problem is solved for a given first-stage decision x and a realization ω of the random vector $\tilde{\omega}$. The second-stage value function is often referred to as the recourse function. Note that the first-stage decision x affects only the right-hand side of the second-stage problem. In general, any of the second-stage parameters can be affected by $\tilde{\omega}$; however, we restrict our attention to problems where only the right-hand side element (i.e., ξ and C) and variable bounds ($\underline{y}$ and $\bar{y}$) depend on $\tilde{\omega}$.

When the support of Ω is finite, a two-stage stochastic programming (2-SP) problem can be reformulated as a deterministic extensive scenario problem and solved using off-the-shelf solvers. However, when the number of scenarios is large, such an endeavor is computationally prohibitive. Most algorithmic advances in stochastic programming (SP), particularly from a computational point of view, are targeted at two-stage stochastic linear programming (2-SLP) problems where the constraints and objective functions associated with the first-stage and the second-stage optimization problems are linear. These advances exploit the decomposability of 2-SLP problems and can be classified into two primary groups: stage decomposition, exemplified by the L-shaped method [29], and scenario decomposition [25].

Similar stage and scenario decomposition approaches can be adopted for 2-SQP problems. For instance, [24] tackles a 2-SQP problem by dualizing the problem, approximating the dual by a sequence of lower dimensional QP subproblems, and establishing that the optimal solutions sequence converges. In [3], a Newton-type method is proposed for 2-SQP. In [21], the authors extend the Newton-type method to exploit the decomposability of 2-SQP, particularly in computing the Newton directions. While these approaches address problems with finite support, [2] introduces an inexact Newton method and combines it with stochastic decomposition to address 2-SQP problems with continuous support. For the same class of problems, [20] establish the sublinear convergence rate of stochastic decomposition.

To address the computational difficulty of cutting-plane methods, the inclusion of a quadratic regularization term is suggested. The resulting algorithm is often referred to as the proximal bundle algorithm. Introduced for deterministic convex programs by [18], the proximal bundle algorithm was adopted for SP problems in [26]. Inexact variants of the proximal bundle algorithm for deterministic convex programs have been studied in [7] and [17]. In SP, the inexact variants have been studied in [23], albeit only for 2-SLP. The regularized stochastic decomposition in [13] can also be viewed as an inexact proximal bundle algorithm, where the inexactness is due to its inherent stochastic design. While the original algorithm is for 2-SLP, recently, [20] extended it to 2-SQP.

Our algorithm designs rely on efficiently solving QP problems that differ only along a few model parameters. Active-set methods, along with interior-point methods, are the main methods to solve the QP problem with equality and inequality constraints. The active set methods reduce the size of the problem by identifying and excluding constraints that are not active in the current iteration. Various forms of active-set methods are presented in the literature based on the convexity properties of QP [10], including primal, dual, and primal-dual active-set (PDAS) methods. Primal and dual active-set methods preserve the feasibility of the primal and dual problems, respectively [22]. Regarding PDAS, [15] introduces a method where each iteration involves a partition of variable indices into an active set and an inactive set. The process results in a distinctive solution through a reduced KKT system. The partition is iteratively adjusted until the obtained solution satisfies both primal and dual feasibility conditions. This algorithm is presented for bound-constrained QP problems, and it is shown to cycle on more general convex QP problems. The algorithm outlined in [15] is the foundation for the methodologies presented in the following two works. In [6], an auxiliary index set is introduced alongside the active-set estimate. This auxiliary set is designated to contain the indices of variables for which bounds need to be explicitly enforced in a given iteration. This approach is globally convergent for strictly convex QP problems. In [5], convergence of the PDAS method is presented under certain assumptions on the structure of the QP problems even when the reduced KKT system is solved inexactly.

1.1 Contributions

In light of the current literature on decomposition methods for 2-SQP, this paper makes three main contributions. A good understanding of the sensitivity analysis and solution properties of the QP problems is essential to develop effective decomposition strategies. In this regard, we first analyze the solution properties of QP problems with specific right-hand sides. We establish that for small perturbations of the right-hand side within a polyhedron, the optimal partition of the variables into active and inactive sets does not change. Furthermore, we show that the partition that yields a feasible dual solution, which is necessary to construct the approximation in bundle algorithms, remains the same for perturbations within a larger polyhedron. We adopt a PDAS method that, starting with an arbitrary partition, recovers a feasible dual solution in a computationally efficient manner.

Secondly, we present two variants of inexact proximal bundle algorithms that rely on efficiently reutilizing solution information gathered in earlier iterations to deal with

second-stage subproblems. The first method, referred to as a dual-based method, collects the dual solutions associated with second-stage problems in a set and generates the suboptimal solution based on this set. On the other hand, the second method called the partition-based method, involves collecting the optimal partitions obtained from previous iterations in a set, and it uses the PDAS method over that set to generate a suboptimal solution. We establish asymptotic convergence of both the inexact methods. We also demonstrate that, when compared to the exact proximal bundle algorithm, the inexact variants generate solutions of comparable quality and significantly reduce computational time.

In addition to algorithm design and convergence analyses, a supplementary contribution of this work is the general-purpose implementation of the variants of proximal bundle algorithms in the C programming language. Our implementation leverages the problem features and dedicated data structures to manage memory assignments efficiently. This carefully designed implementation minimizes computational overhead by storing and reusing frequently accessed information (dual solutions, partitions, etc.) and allows us to reduce the overall computational workload.

1.2 Organization

We organize the paper as follows. In Section 2, we analyze the solution properties of QP problems and relate them to the notion of variable partitions. We also adopt a PDAS method to recover solutions in a computationally efficient manner. In Section 3, we introduce the inexact proximal bundle algorithms and present their asymptotic analyses. We present the results from the numerical experiments in Section 4. Finally, in Section 5 we discuss the conclusions drawn from our study and propose potential areas for future research. Proofs for most of the results are included in Appendix A.

2 Solution Properties of QP Problems

Our aim is to develop decomposition-based solution algorithms for 2-SQP and to do so; we first analyze QP problems in this section. We then apply this analysis in §3 to present the solution methods for 2-SQP. For the purposes of our discussion, we consider the following QP problem:

$$\begin{aligned} \min \;\; & g_P(y; x, \omega) := \frac{1}{2} y^\top Q y + d^\top y \\ \text{subject to } & Dy = \rho, \; \underline{y} \leq y \leq \bar{y}. \end{aligned} \tag{2}$$

The objective function $g_P(y; x, \omega)$ is a linear-quadratic function that is convex if Q is a positive semidefinite matrix. We denote the set of variable and constraint indices as $\mathsf{N} := \{1, \ldots, n_2\}$ and $\mathsf{M} := \{1, \ldots, m_2\}$, respectively. It is worth noting that the QP problem presented above resembles the second-stage problem in (1b). For ease of notation, we assume that C and $\underline{y}$ are independent of ω, although the subsequent outcomes remain applicable even if they are dependent. The problem (2) is obtained by substituting the

right-hand side of the equality constraints with $\rho = \xi(\omega) - Cx$ and the upper bounds with $\bar{y} = \bar{y}(\omega)$.

An important property of (2) is that the feasible region of the dual problem is independent of the right-hand side parameters. The dual of QP problem is given by:

$$\begin{aligned} \max\ & g_D(\pi; x, \omega) = -\frac{1}{2}u^\top Q u + \rho^T \lambda + \underline{y}^T \nu - \bar{y}^T \mu \\ \text{subject to } & D^T\lambda + \nu - \mu - Qu = d,\ \nu, \mu \geq 0. \end{aligned} \tag{3}$$

The above dual is the type II form of QP dual given in [8]. According to [8], if y^* is the optimal solution of (2), there exists an optimal solution for (3) such that $u^* = y^*$. Therefore, we use $u = y$ in the remainder of this paper when referring to the dual problem. Let λ denote the dual variables (Lagrange multiplier) associated with the equality constraints, and μ and ν are dual variables associated with upper and lower bounds, respectively. We collectively denote the dual solution vector $\pi \coloneqq (y, \lambda, \nu, \mu)$.

2.1 Solution Recovery using Decomposed KKT System

In addition to the aforementioned property, analyzing the effect of the QP problem's parameters on the first-order KKT optimality conditions reveals additional useful properties. For the QP in (2), at optimality, the primal variable y and the dual variables (λ, μ, ν) satisfy KKT conditions that are given as:

$$Qy + d + \mu - \nu - D^T\lambda = 0, \tag{4a}$$
$$Dy = \rho, ,\ \underline{y} \leq y \leq \bar{y}, \tag{4b}$$
$$0 \leq \mu \perp (\bar{y} - y) \geq 0, \tag{4c}$$
$$0 \leq \nu \perp (y - \underline{y}) \geq 0. \tag{4d}$$

A primal solution y to (2) has elements either at their lower-bound ($y_i = \underline{y}_i$), upper-bound ($y_i = \bar{y}_i$), or in the interior of the interval ($\underline{y}_i < y_i < \bar{y}_i$). To distinguish these elements, we associate with a primal solution mutually exclusive and exhaustive subsets of indices denoted by L, U, and I that correspond to elements at lower-bound, upper-bound, and interior (inactive), respectively. That is

$$\mathsf{L} := \{i \mid y_i = \underline{y}_i, i \in \mathsf{N}\};\ \mathsf{U} := \{i \mid y_i = \bar{y}_i, i \in \mathsf{N}\};\ \mathsf{I} := \{i \mid \underline{y}_i < y_i < \bar{y}_i, i \in \mathsf{N}\}.$$

We refer to set $\mathsf{A} = \mathsf{L} \cup \mathsf{U}$ as the *active* set and I as the *inactive* set. Since the mutually exclusive and exhaustive sets represent the partition of the index set N, we refer to the tuple $P = (\mathsf{L}, \mathsf{U}, \mathsf{I})$ as a *partition*. Specifically, we will refer to the partition associated with a solution that satisfies the KKT conditions in (4) as the *optimal partition*. In this paper, we use two index sets as subscripts to specify a sub-matrix of a matrix. For example, Q_{MB} indicates a sub-matrix of Q formed by rows and columns indexed by M and B, respectively.

The concept of partition is a useful tool in analyzing the KKT system. For a given partition P, the primal variable y, the dual variables (λ, ν, μ), and the problem parameters

(Q, D, d) can be decomposed. Using the decomposed elements, (4a) and (4b) can be combined and written as the following system of equations:

$$\begin{bmatrix} Q_{\mathsf{LL}} & Q_{\mathsf{LU}} & Q_{\mathsf{LI}} & D_{\mathsf{ML}}^T \\ Q_{\mathsf{UL}} & Q_{\mathsf{UU}} & Q_{\mathsf{UI}} & D_{\mathsf{MU}}^T \\ Q_{\mathsf{IL}} & Q_{\mathsf{IU}} & Q_{\mathsf{II}} & D_{\mathsf{MI}}^T \\ D_{\mathsf{ML}} & D_{\mathsf{MU}} & D_{\mathsf{MI}} & 0 \end{bmatrix} \begin{bmatrix} y_{\mathsf{L}} \\ y_{\mathsf{U}} \\ y_{\mathsf{I}} \\ \lambda \end{bmatrix} + \begin{bmatrix} \mu_{\mathsf{L}} \\ \mu_{\mathsf{U}} \\ \mu_{\mathsf{I}} \\ 0 \end{bmatrix} - \begin{bmatrix} \nu_{\mathsf{L}} \\ \nu_{\mathsf{U}} \\ \nu_{\mathsf{I}} \\ 0 \end{bmatrix} + \begin{bmatrix} d_{\mathsf{L}} \\ d_{\mathsf{U}} \\ d_{\mathsf{I}} \\ -\rho \end{bmatrix} = 0. \tag{5}$$

From the definitions of sets L and U, and complementary slackness in (4), we have:

$$y_{\mathsf{L}} = \underline{y}_{\mathsf{L}}; \quad y_{\mathsf{U}} = \bar{y}_{\mathsf{U}}; \quad \mu_{\mathsf{L}} = \nu_{\mathsf{U}} = \mu_{\mathsf{I}} = \nu_{\mathsf{I}} = 0. \tag{6}$$

Substituting the above in (5), we obtain the reduced KKT system:

$$\begin{bmatrix} Q_{\mathsf{LL}} \\ Q_{\mathsf{UL}} \\ Q_{\mathsf{IL}} \\ D_{\mathsf{ML}} \end{bmatrix} \underline{y}_{\mathsf{L}} + \begin{bmatrix} Q_{\mathsf{LU}} \\ Q_{\mathsf{UU}} \\ Q_{\mathsf{IU}} \\ D_{\mathsf{MU}} \end{bmatrix} \bar{y}_{\mathsf{U}} + \begin{bmatrix} Q_{\mathsf{LI}} & D_{\mathsf{ML}}^T \\ Q_{\mathsf{UI}} & D_{\mathsf{MU}}^T \\ Q_{\mathsf{II}} & D_{\mathsf{MI}}^T \\ D_{\mathsf{MI}} & 0 \end{bmatrix} \begin{bmatrix} y_{\mathsf{I}} \\ \lambda \end{bmatrix} + \begin{bmatrix} -\nu_{\mathsf{L}} \\ \mu_{\mathsf{U}} \\ 0_{\mathsf{I}} \\ 0_{\mathsf{M}} \end{bmatrix} + \begin{bmatrix} d_{\mathsf{L}} \\ d_{\mathsf{U}} \\ d_{\mathsf{I}} \\ -\rho \end{bmatrix} = 0. \tag{7}$$

The following result captures the behavior of the solutions of the above KKT system.

Proposition 2.1. *Let $P = (\mathsf{L}, \mathsf{U}, \mathsf{I})$ be the optimal partition of (2) for a given right-hand side and upper bound $(\rho, \bar{y})$. Then,*

(a) there exists a polyhedron S_1, such that the partition P can be used to construct a solution (y, λ, μ, ν) that is feasible to the dual problem in (3) associated with any other right-hand side and upper bound $(\rho', \bar{y}') \in S_1$.

(b) Furthermore, there exists a polyhedron $S_2 \subseteq S_1$ such that P remains an optimal partition for (2) associated with every right-hand side and upper bound $(\rho', \bar{y}') \in S_2$.

We would like to emphasize that the property identified in Proposition 2.1 plays a crucial role in the design of our algorithm. However, it is worth noting that we do not aim to explicitly construct the polyhedra S_1 and S_2 in our algorithm. Instead, we generate matrices:

$$M = \begin{bmatrix} Q_{\mathsf{II}} & -D_{\mathsf{MI}}^T \\ D_{\mathsf{MI}} & 0 \end{bmatrix}^{-1}; \qquad W = -\begin{bmatrix} Q_{\mathsf{II}} & -D_{\mathsf{MI}}^T \\ D_{\mathsf{MI}} & 0 \end{bmatrix}^{-1} \begin{bmatrix} Q_{\mathsf{IU}} & 0_{\mathsf{IM}} \\ D_{\mathsf{MU}} & -I \end{bmatrix}, \tag{8}$$

and

$$T = -\begin{bmatrix} Q_{\mathsf{LU}} & 0_{\mathsf{LM}} \\ -Q_{\mathsf{UU}} & 0_{\mathsf{UM}} \end{bmatrix} + \begin{bmatrix} Q_{\mathsf{LI}} & D_{\mathsf{ML}}^T \\ -Q_{\mathsf{UI}} & -D_{\mathsf{MU}}^T \end{bmatrix} W, \tag{9}$$

for every partition that the algorithm discovers; see Appendix A regarding the construction of the above matrices. These matrices can be utilized to compute dual feasible solutions for any perturbed problem with right-hand side and upper bound $(\rho', \bar{y}') \in S_1$, by means of an affine transformation of the perturbation and the solution of the original problem. The details of these computations are presented in the proof of Proposition 2.1, and are summarized in the following corollary.

Corollary 2.2. *Let P and (y, λ, ν, μ) be the optimal partition and optimal solution of problem (2) with right-hand side and upper bound $(\rho, \bar{y})$, respectively. Then, for any $(\rho', \bar{y}') \in S_1$, a feasible dual solution is derived from partition P as:*

$$\begin{bmatrix} y'_{\mathsf{I}} \\ \lambda' \\ \nu'_{\mathsf{L}} \\ \mu'_{\mathsf{U}} \end{bmatrix} = \begin{bmatrix} y_{\mathsf{I}} \\ \lambda \\ \nu_{\mathsf{L}} \\ \mu_{\mathsf{U}} \end{bmatrix} + \begin{bmatrix} W \\ T \end{bmatrix} \begin{bmatrix} \Delta\bar{y}_{\mathsf{U}} \\ \Delta\rho \end{bmatrix}, \tag{10}$$

and $y'_{\mathsf{U}} = \bar{y}_{\mathsf{U}} + \Delta\bar{y}_{\mathsf{U}}$; $y'_{\mathsf{L}} = \underline{y}_{\mathsf{L}}$; $\mu'_{\mathsf{L}} = 0$; $\nu'_{\mathsf{U}} = 0$; $\mu'_{\mathsf{I}} = 0$; $\nu'_{\mathsf{I}} = 0$.

If we solve problem (2) to optimality for a particular right-hand side $(\rho, \bar{y})$, then we have access to the corresponding optimal partition P. The above results indicate that this partition can be used to recover a solution for a perturbed program with right-hand side $(\rho', \bar{y}')$. If the solution satisfies (4c) and (4d), then we have a feasible dual solution. If the solution further satisfies bound constraints, then P serves as an optimal partition for the perturbed program. However, if either (4c) or (4d) is not met for the given right-hand side values $(\rho', \bar{y}')$, which is equivalent to $(\rho', \bar{y}') \notin S_1$, we recover a feasible dual solution by employing the PDAS method. We elaborate on the details in the subsequent section.

2.2 Solution Recovery using PDAS Method

The PDAS method is a versatile algorithm used in various optimization problems, particularly those with inequality constraints. The PDAS methods provide an efficient approach to identifying optimal partitions for deterministic optimization problems [5, 6, 16, 22]. In these methods, the index sets are updated based on information available at the current solution. For our purpose, it suffices to identify a feasible dual solution as opposed to an optimal solution. Therefore, for a partition that yields an infeasible dual solution, we employ the PDAS method to recover a partition that yields a feasible dual solution. To achieve this, we develop four different versions of the PDAS method suggested in [15]. We begin by explaining how PDAS is employed according to [15], followed by a discussion of the four algorithmic variations considered in our investigation.

Algorithm 1 presents the PDAS method to find a feasible dual solution for a given right-hand side values $(\rho', \bar{y}')$. This algorithm utilizes the initial partition $P^0 = (\mathsf{L}^0, \mathsf{U}^0, \mathsf{I}^0)$, and obtains its associated solution π^0 from equation (10). We proceed if this solution does not satisfy the dual feasibility condition for the second-stage problem. Sets V_P^0 and V_D^0 are determined based on the initial solution π^0, signifying the primal and dual variables that do not currently satisfy bound constraints and non-negativity constraints, respectively. Additionally, we initialize the iteration counter as $m = 0$.

At the beginning of iteration m, we first update the iteration counter $m = m + 1$ and select a set $V'_D \subseteq V_D^{m-1}$ and a subset $V'_P \subseteq V_P^{m-1}$. This selection is based on one of the four strategies we discuss later. Next, observe that variables within the subset V'_P violate either their upper or lower bounds. We partition V'_P into two distinct subsets: V'_{P_L} that encompasses variables that are smaller than their lower bounds and V'_{P_U} that comprises variables that exceed their upper bounds. Similarly, observe that variables within the subset V'_D are either active at their lower or upper bound. We divide this subset into two

Algorithm 1 PDAS

Input: $P^0 = (\mathsf{L}^0, \mathsf{U}^0, \mathsf{I}^0)$.
Initialization: Compute π^0 using P^0; Identify V_P^0 and V_D^0; Set $m \leftarrow 0$.
while $V_D^m \neq \emptyset$ **do**
  $m \leftarrow m + 1$.
  Select subsets $V_D' \subseteq V_D^{m-1}$ and $V_P' \subseteq V_P^{m-1}$ from an elemental selection approach.
  Identify subsets V_{PL}' and V_{PU}' from V_P' and subsets V_{DL}' and V_{DU}' from V_D'.
  Perform the activation and inactivation phases to obtain the updated index sets as shown in (11a).
  With the updated partition $P^m = (\mathsf{L}^m, \mathsf{U}^m, \mathsf{I}^m)$, solve the reduced KKT system to obtain π^m and V_P^m and V_D^m.
end while
Return: P^m and π^m.

separate subsets: V_{DL}' that consists of variables that are active at their lower bound and V_{DU}' that encompasses variables that are active at their upper bounds.

Following that, to modify the partition, we go through two phases, referred to as activation and inactivation phases. In the activation phase, we activate the elements in V_{PL}' at their lower bounds and the elements in V_{PU}' at their upper bounds. In the inactivation phase, we deactivate the variables associated with infeasible dual solutions. We achieve this by moving the elements of V_D' from U^{m-1} or L^{m-1} to the set I^m to obtain the updated index sets. We capture these updates in the following:

$$\mathsf{U}^m \leftarrow (\mathsf{U}^{m-1} \setminus V_{DU}') \cup V_{PU}', \tag{11a}$$
$$\mathsf{L}^m \leftarrow (\mathsf{L}^{m-1} \setminus V_{DL}') \cup V_{PL}', \tag{11b}$$
$$\mathsf{I}^m \leftarrow (\mathsf{I}^{m-1} \setminus V_P') \cup V_D' \tag{11c}$$

Finally, based on the modified partition $P^m = (\mathsf{L}^m, \mathsf{U}^m, \mathsf{I}^m)$, we solve the reduced KKT system (7) for $(\rho', \bar{y}')$ to achieve a new solution π^m. We also identify new sets of primal and dual infeasible variables, V_P^m and V_D^m, respectively. If the new solution is dual feasible, that is, $V_D^m \neq \emptyset$, then the method terminates. Otherwise, we repeat the above steps.

In the context of the PDAS method, finite convergence is not generally guaranteed. Typically, Previous works, e.g., [5] and [15], each impose restrictive assumptions on QP problems to ensure convergence. The paper [15] addresses this by examining problems constrained exclusively by the lower and upper bounds on decision variables. Likewise, the study [5] investigates scenarios where variables are exclusively bounded from above. In this context, they impose rigorous conditions concerning the reduced Hessian in terms of the upper-bounded variables. These conditions must hold for every potential partition to ensure convergence. Additionally, [6] discusses selection approaches that involve defining an auxiliary set and solving a QP problem in each iteration to ensure convergence. These studies collectively underscore the PDAS method's reliance on particular problem structures and assumptions for assured finite convergence. Based on the insight we build from these studies, we introduce four approaches for selecting the subsets V_P' and V_D':

- Single-Swap: Randomly choose an element $v_P \in V_P^{m-1}$ and an element $v_D \in V_D^{m-1}$. Set $V_P' \leftarrow \{v_P\}$ and $V_D' \leftarrow \{v_D\}$.

- Aggressive-Dual: Randomly select one element v_P from V_P^{m-1}, resulting in $V_P' \leftarrow \{v_P\}$, and set $V_D' \leftarrow V_D^{m-1}$.
- Aggressive-Primal: Randomly pick one element v_D from V_D^{m-1}, leading to $V_D' = \{v_D\}$, and set $V_P' \leftarrow V_P^{m-1}$.
- Aggressive-Primal-Dual: Set $V_P' \leftarrow V_P^{m-1}$ and $V_D' \leftarrow V_D^{m-1}$.

These approaches effectively showcase a range of options, spanning from aggressive to conservative. In Section 4, we undertake a comprehensive comparison of these four approaches to assess their performance in identifying a feasible dual solution. Our evaluation is based on criteria such as the frequency of encountered cycles, runtime efficiency, and the quality of the obtained solution.

3 Inexact Proximal Bundle Algorithms for 2-SQP

In this section, we incorporate the PDAS method in decomposition-based proximal bundle algorithms for solving the 2-SQP problem in (1). For these algorithms, we make the following assumptions regarding (1).

(A1) The first-stage feasible region $\mathcal{X}$ is convex and compact.

(A2) The matrices P and Q are positive semidefinite and positive definite, respectively. Additionally, the matrix D is full rank.

(A3) The 2-SQP satisfies relatively complete recourse, and the recourse value is bounded from below by $L > -\infty$.

(A4) The randomness affects right-hand side elements or variable bounds, and the support Ω is a finite set.

These assumptions are commonly made in SP literature. Assumption (A2) implies that the 2-SQP is a convex program. Furthermore, it indicates that the second stage is strictly convex and has a unique solution. Assumption (A3) ensures that for all $x \in \mathcal{X}$ and $\omega \in \Omega$, the second-stage problem (1b) is feasible and its dual feasible region is bounded. Finally, assumption (A4) allows us to write the expectation in (1a) as a finite sum. When a finite sample, say $\widehat{\Omega}$, either generated using computer simulations or from historical data, is used to approximate Ω, the problem reduces to a *sample average approximation* problem. The algorithms presented in this section apply to such problems as well. Two-stage problems with a deterministic recourse matrix D are said to have fixed recourse, and with appropriate pre-processing, we can ensure that the problem has a full-rank recourse matrix. The exact proximal bundle algorithms are well-studied both for deterministic and stochastic programs. In the case of 2-SP problems, the exact proximal bundle algorithm is commonly referred to as the regularized L-shaped method. The inexact variants we present here share the overall structure of the proximal bundle algorithms and differ only in how we generate the affine functions that constitute the bundle. We refer the reader to [1] or [27] for a detailed exposition of the proximal bundle algorithm applied to 2-SP

Algorithm 2 Proximal-bundle algorithm for 2-SQP

1: **Input:** 2-SQP problem parameters, $\gamma \in (0,1)$, and $\sigma \in (0,1)$.
2: **Initialization:** $k \leftarrow 0$, $J^0 = \{(L,0)\}$, $f^0(x) = \frac{1}{2}x^\top Px + c^\top x + L$, and $w^0 \in \mathcal{X}$.
3: **while** the stopping rule is not satisfied **do**
4: $k \leftarrow k+1$
5: *Candidate update:* identify a candidate solution:

$$x^k \in \arg\min\{f_\sigma^{k-1}(x) \coloneqq f^{k-1}(x) + \frac{\sigma}{2}\|x - w^{k-1}\| \;\big|\; x \in \mathcal{X}\}. \tag{12}$$

6: *Minorant generation:* generate a minorant (α^k, β^k) using an exact or inexact bundle procedure.
7: *Bundle update:* update the bundle:

$$J^k = J^{k-1} \cup (\alpha^k, \beta^k).$$

8: *Approximation update:* Obtain an updated approximation:

$$f^k(x) = \frac{1}{2}x^\top Px + c^\top x + \max_{(\alpha^j,\beta^j)\in J^k}(\alpha^j + {\beta^j}^\top x). \tag{13}$$

9: **if** $f^k(x^k) - f^k(w^{k-1}) \leq \gamma(f^{k-1}(x^k) - f^{k-1}(w^{k-1}))$ **then**
10: *Incumbent update:* $w^k \leftarrow x^k$,
11: **else**
12: $w^k \leftarrow w^{k-1}$.
13: **end if**
14: **end while**
15: **Return:** The optimal solution $x^* \leftarrow w^k$.

problems. Here, we briefly overview the algorithm and discuss some essential elements. The steps are presented in Algorithm 2.

The bundle algorithms utilize a set of affine functions to approximate the expected recourse function, resulting in a piecewise affine convex lower envelope for the expectation-valued objective function in (1a). An algorithm iteration, indexed by k, begins by identifying a candidate solution x^k, which we obtain by solving a proximal (or regularized) master problem (see Line 5). We construct a minorant at the candidate solution using either the exact or inexact procedure (introduced later in this section). We denote by (α^k, β^k) the coefficients of the minorant that we construct in the current iteration. We add this minorant to the existing bundle to obtain the updated bundle (see Line 7). Using the updated bundle, we update the approximation of the first-stage objective function, which we subsequently use to update the incumbent solution in Line 9. This completes the iteration.

A few observations about the proximal bundle algorithm are in order. The construction of the minorants utilizes the dual solutions of the second-stage program. Let $\pi^k(\omega)$ be the optimal dual solution obtained by solving the second-stage program (1b) with

input (x^k, ω). Due to weak duality, we have:

$$\begin{aligned} h(x,\omega) \geq & -\frac{1}{2}(y^k)^\top Q y^k + (\xi(\omega) - Cx)^\top \lambda^k + \underline{y}^\top \nu^k - \bar{y}^\top \mu^k \\ = & \underbrace{-\frac{1}{2}(y^k)^\top Q y^k + \xi(\omega)^\top \lambda^k + \underline{y}^\top \nu^k - \bar{y}^\top \mu^k}_{:=\alpha^k(\omega)} + \underbrace{\big(-(\lambda^k)^\top C\big)}_{:=\beta^k(\omega)} x, \ \forall x \in \mathcal{X}. \end{aligned} \tag{14}$$

The statement above holds with strict equality when $x = x^k$ as a consequence of strong duality. In this case, we achieve an exact or supporting hyperplane for $h(x, \omega)$. However, when we use only a feasible solution, the inequality (14) still applies, but the resulting affine function may not be supporting even at $x = x^k$. Consequently, we obtain an inexact (lower-bounding) approximation.

Typically, when the bundle algorithms are applied for 2-SP, we solve all second-stage problems to optimality to compute the coefficients of the minorants as outlined in Algorithm 3. We refer to this procedure for generating minorants as the *exact procedure*. Consequently, we obtain exact/supporting hyperplanes to individual recourse functions

Algorithm 3 Exact procedure for generating minorants

1: **Input:** Candidate solution x^k and the sample set Ω.
2: **for** $\omega \in \Omega$ **do**
3: Solve the second-stage program and obtain the optimal solution π^k.
4: Calculate coefficients $(\alpha^k(\omega), \beta^k(\omega))$ from (14).
5: **end for**
6: Aggregate the coefficients of the affine function as:

$$\alpha^k = \sum_{\omega \in \Omega} p(\omega)\alpha^k(\omega), \quad \beta^k = \sum_{\omega \in \Omega} p(\omega)\beta^k(\omega). \tag{15}$$

7: **return** (α^k, β^k).

$h(x, \omega)$ and the resulting minorant satisfies $\alpha^k + (\beta^k)^\top x^k = \mathbb{E}[h(x^k, \tilde{\omega})]$, for every $k \geq 1$. Moreover, the approximation satisfies $f^k(x^k) = f(x^k)$ and $f^k(x) \leq f(x)$ for all $x \in \mathcal{X}$ and $k \geq 1$ (see (1a) and (13)). Over the iterations, the approximations are monotonically increasing, that is, $f^{k-1}(x) \leq f^k(x) \leq f^{k+1}(x) \leq \ldots \leq f(x)$ for all $x \in \mathcal{X}$. These properties play a vital role in showing the algorithm's convergence when we adopt the exact procedure for generating the minorants (see, for example, chapter 3 in [27]).

Secondly, the proximal bundle algorithm maintains two sequences of solutions: a candidate sequence $\{x^k\}$ and an incumbent sequence $\{w^k\} \subseteq \{x^k\}$. The criterion used to identify the incumbent solutions plays a critical role in the convergence and computational performance of the algorithm. Notice that when we use the exact procedure, we can access the function value $f(x)$ at the candidate solution and the incumbent solution. This is because at the candidate solution we have $f^k(x^k) = f(x^k)$. Since the incumbent sequence is a subsequence of candidate solutions, we must have a $j < k$ such that $w^{k-1} = x^j$. Since we retain all the previously generated minorants in the bundle J^k, we have $f^k(w^{k-1}) = f^j(x^j) = f(x^j)$. Consequently, the criterion we use to assess whether the current candidate solution can be accepted as an incumbent solution (in Line 9) reduces

to the descent condition: $f(x^k) - f(w^{k-1}) \leq \gamma(f^{k-1}(x^k) - f(w^{k-1}))$. The reader may be familiar with this descent criterion as it is more common for the proximal bundle algorithm (see, for example, [19] and [26]). However, it is not possible to assess such a criterion when only inexact function values are available. Therefore, we assess a criterion using only the inexact function value f^k computed at the current candidate and incumbent solutions. A similar criterion was previously employed in the stochastic decomposition algorithm where coefficients of the minorants are stochastic in nature; see [13].

Thirdly, how we manage the minorant bundle is also vital in determining the behavior of the algorithms. Notice that we add a new minorant to the bundle in every iteration (Step 7 of Algorithm 2). While we retain all previously generated minorants in the bundle, it is possible to remove minorants generated in earlier iterations based on whether they contribute to the function approximation at the solutions where the assessment is being made (specifically, candidate and incumbent solution). We assess this contribution by examining the optimal dual solutions, denoted by ζ^j, corresponding to an element $j \in J^{k-1}$ obtained after solving (12). A bundle, given by $J^k = \{j \mid \zeta^j > 0\} \cup (\alpha^k, \beta^k) \cup \{\hat{\alpha}, \hat{\beta}\}$ serves this purpose. Here, $(\hat{\alpha}, \hat{\beta})$ denotes the minorant generated at the incumbent solution, which was generated in an earlier iteration, that is, $(\hat{\alpha}, \hat{\beta}) = (\alpha^j, \beta^j)$ for some $j < k$. When we employ a bundle comprising only the active minorants, convergence analysis of the algorithms adds additional considerations. Since we focus on studying the consequence of introducing inexactness in minorant generation, we maintain a full bundle.

The advantages of employing the exact procedure come at the significant computational cost of solving second-stage programs for all observations in Ω. This is particularly the case when dealing with large-scale problems in the second stage or when we have an exorbitantly large number of possible scenarios. To alleviate this bottleneck, we introduce two inexact procedures for constructing the approximations used in the bundle algorithm. In these inexact procedures, we solve subproblems only for a subset of observations and use an appropriate approximate solution for the remaining observations. This results in inexactness both in the function value and subgradient. Moreover, in our algorithms, the level of inexactness is unknown but bounded and vanishes as the algorithm progresses.

3.1 Dual-based Inexact Procedure

Due to assumption (A4), the set of feasible dual solutions is not affected by either the first-stage decision variable or the observation, see (3). We utilize this fundamental characteristic of the 2-SQP problems in the first inexact procedure for generating the minorants. We describe this procedure in Algorithm 4.

In the dual-based procedure, we maintain a collection of previously encountered dual solutions. At the beginning of iteration k, we denote this collection by Π^{k-1}. To generate a new minorant at candidate solution x^k, we select a subset Ω^k of observations and solve the subproblems corresponding to only observations $\omega \in \Omega^k$, i.e., with input (x^k, ω) in (1b). We append the optimal dual solutions obtained by solving these subproblems to the set Π^{k-1}, thus obtaining an updated set Π^k. For the remaining observations $\omega \in \Omega \setminus \Omega^k$,

Algorithm 4 Dual-based inexact procedure for generating minorant

Input: Candidate solution x^k, sample set Ω, collection of dual solutions Π^{k-1}.
Randomly generate a subset $\Omega^k \subset \Omega$.
for $\omega \in \Omega^k$ **do**
  Solve (1b) for (x^k, ω) and obtain the optimal dual solution $\pi(\omega)$.
  Calculate $(\alpha^k(\omega), \beta^k(\omega))$ from (14).
end for
Update the collection: $\Pi^k = \Pi^{k-1} \cup \{\pi(\omega) | \omega \in \Omega^k\}$
for $\omega \in \Omega \setminus \Omega^k$ **do**
  Using the argmax step in (16), identify an approximate dual solution.
  Calculate $(\alpha^k(\omega), \beta^k(\omega))$ from (14).
end for
Calculate the affine function coefficients from (15).
Return: (α^k, β^k) and Π^k.

we obtain a suboptimal dual solution by utilizing an "argmax" step:

$$\pi(\omega) \in \arg\max \big\{ \frac{1}{2} y^\top Q y + (\xi(\omega) - Cx^k)^\top \lambda + \underline{y}^T \nu - \bar{y}(\omega)^\top \mu \ | \pi \in \Pi^k \big\}. \tag{16}$$

This step seeks to identify a solution from the set Π^k that yields the best objective function value for the dual of the second-stage problem. Next, by utilizing the dual solutions, whether optimal or approximate, we compute the coefficients of the lower bounding minorant using the calculations shown in (14). When the dual-based inexact procedure is embedded in the proximal bundle algorithm in Algorithm 2, we initialize the collection $\Pi^0 = \emptyset$ and maintain the collection through the course of the algorithm. We refer the algorithm as dual-based proximal bundle algorithm.

The idea of reutilizing previously encountered solutions has been effectively utilized in SP. The earliest work that we are aware of is [30] for 2-SLP, where the author proposes bunching observations that lead to the same optimal basis of the second-stage linear programs and using these bunches to speed up the computations of minorant generation in the L-shaped method. Using a similar design principle, [31] proposes an inexact variant of the L-shaped method, and subsequently, [23] proposes an inexact proximal bundle method for 2-SLP. Reutilizing optimal dual solutions for 2-SLP with right-hand side uncertainty and the optimal dual basis for 2-SLP with random cost coefficients drives the computational efficiency of the stochastic decomposition algorithm; see [13] and [11], respectively.

3.2 Partition-based Inexact Procedure

While the dual-based inexact procedure reduces the computational effort required to solve the second-stage problems, it inadvertently leads to an escalation in the size of the dual set Π^k through successive iterations. This increase in size consequently renders the argmax step computationally burdensome, as it necessitates the examination of every element within the set Π^k during each iteration k. To avoid this drawback, we utilize the result of Proposition 2.1 to design an inexact procedure to generate the minorant for the proximal bundle algorithm. We summarize this procedure in Algorithm 5.

Algorithm 5 Partition-based inexact procedure for generating minorant

Input: Candidate solution x_k, sample set Ω, collection of partitions $\mathcal{P}^{k-1}$.
Initialization: $\Pi^k \leftarrow \emptyset$.
Randomly generate a subset $\Omega^k \subset \Omega$
for $\omega \in \Omega^k$ **do**
  Solve (1b) for (x^k, ω) and obtain the optimal partition $P(\omega)$.
  Calculate $(\alpha(\omega), \beta(\omega))$ from (14).
end for
Update the collection: $\mathcal{P}^k \leftarrow \mathcal{P}^{k-1} \cup \{P(\omega) | \omega \in \Omega^k\}$.
for $\omega \in \Omega \diagdown \Omega^k$ **do**
  Apply Algorithm 1 for all $p \in \mathcal{P}^k$ and gather all dual feasible solutions in a set defined as $\Pi^k = \Pi(x, \omega, \mathcal{P}^k)$.
  Use the argmax step in (16) to obtain the approximated solution.
  Calculate $(\alpha(\omega), \beta(\omega))$ from (14).
end for
Calculate affine function coefficients from (15).
Return: (α^k, β^k) and $\mathcal{P}^k$.

In each iteration, we solve second-stage problems for a subset of observations, as in Algorithm 4. However, instead of storing the dual solution, we store their optimal partitions. The input collection of partitions $\mathcal{P}^{k-1}$ is updated using partitions obtained by solving subproblems corresponding to $\omega \in \Omega^k$. For the remaining observations, we use the PDAS method described in Algorithm 1 to construct a feasible dual solution for all partitions $p \in \mathcal{P}^k$. We then invoke the argmax step in (16) to identify the best dual solution. Since there are finitely many partitions, the arg max step can be carried out computationally efficiently. We compute the lower bounding function coefficients using the calculations shown in (14). When the partition-based inexact procedure is embedded in the proximal bundle algorithm in Algorithm 2, we initialize the collection $\mathcal{P}^0 = \emptyset$ and consistently update this collection throughout the algorithm's execution. This algorithm is denoted as the partition-based proximal bundle algorithm. We close this section with the following remarks.

Remark 3.1. This remark addresses the process adopted to select the subset $\Omega^k \subseteq \Omega$. Let $\mathcal{J}_\omega \subseteq \{k = 0, 1, \ldots\}$ be the iterations, when $\omega \in \Omega^k$. The criterion for selecting Ω^k must satisfy $|\mathcal{J}_\omega| = \infty$. This criterion is equivalent to saying that for any $K > 0$, there exists a $j > K$ such that $j \in \mathcal{J}_\omega$. This requirement is satisfied, for instance, in a deterministic process such as the one adopted in the exact procedure where $\Omega^k = \Omega$. Another example is when observations are selected cyclically: $\omega_m \in \Omega^k$ for $k = n|\Omega| + m, n > 0, m = 1, \ldots, |\Omega|$. The requirement is also satisfied when randomization is adopted. For instance, when every observation is selected with fixed nonzero probability p, independently in each iteration. The requirement is satisfied due to the Borel-Cantelli lemma [9].

Remark 3.2. Like the proximal bundle algorithm, the level-decomposition method [19] provides an alternate approach to handle the challenges associated with the classical cutting-plane algorithm. Both these methods rely on an outer approximation of the first-stage objective function and principally differ in how a new candidate solution is identified. Our approaches for generating the piecewise affine functions for the outer approximations also apply to level-decomposition methods.

3.3 Asymptotic Convergence

In this section, we present the asymptotic convergence results for both variants of the inexact proximal bundle algorithm. Recall that $\{x^k\}$ and $\{w^k\}$ denote the sequence of candidate and incumbent solutions generated by the algorithm. The first result establishes the behavior of the first-stage function approximation along a converging sequence of solutions.

Lemma 3.1. *Let $\mathcal{K}$ index an infinite subsequence of candidate solutions $\{x^k\}_{k=1}^{\infty}$ such that $\{x^k\}_{\mathcal{K}} \to \hat{x}$. Then have $\{f^k(x^k)\}_{\mathcal{K}} \to f(\hat{x})$.*

The analysis that leads to the above result is distinct when the procedure for generating the minorants differs. To establish the result in the case when we employ the dual-based procedure, we exploit the fact that the collection of feasible solutions exhibits the following property: $\Pi^k \subseteq \Pi^{k+1} \subseteq \ldots \subseteq \Pi$. This involves leveraging the continuity of the optimal value of (1) with respect to x. On the other hand, for the partition-based method, we rely on $\mathcal{P}^k \subseteq \mathcal{P}^{k+1} \subseteq \ldots \subseteq \mathcal{P}$ and the selection of the best solution obtained from applying PDAS through the argmax step. We establish the result based on Proposition 2.1, where we demonstrated that optimal partition does not change when the solution x^k, and consequently, the right-hand side $\rho = \xi(\omega) - Cx^k$ is within a polyhedron. The proof also establishes uniform convergence of the sequence $\{f^k\}$.

For the next results, the criterion used to update the incumbent solutions plays a vital role. We denote by $\mathcal{N} \subseteq \{0, 1, \ldots\}$ the iterations at which we update the incumbent (referred to as the serious steps in the literature related to proximal bundle methods). We recognize a few quantities that appear frequently in our analysis. We denote by θ^k the optimal value of the first-stage approximate problem (12), i.e.,

$$\theta^k \coloneqq f^{k-1}(x^k) + \frac{\sigma}{2}\|x^k - w^{k-1}\|^2.$$

We capture the perceived improvement at the candidate solution, relative to the current incumbent solution, by the term $\delta^k \coloneqq f^{k-1}(w^{k-1}) - f^{k-1}(x^k)$. Since w^{k-1} is a feasible solution to the master problem, it follows that θ^k is bounded above by $f^{k-1}(w^{k-1})$, and in turn, by $f(w^{k-1})$. Optimality of x^k implies that

$$\delta^k \geq \frac{\sigma}{2}||x^j - w^{k-1}|| \geq 0. \tag{17}$$

The next result establishes the asymptotic behavior of the sequence $\{\delta^k\}$.

Lemma 3.2. *There exists a subsequence $\mathcal{K}^* \subseteq \mathcal{N}$ such that $\{\delta^k\}_{\mathcal{K}^*} \to 0$.*

To prove the above result, we examine the case where the incumbent solution is updated a finite number of times, resulting in the entire sequence of δ^k converging to zero. We present this proof in Appendix A and focus on the second case, where the incumbent solution is updated infinitely.

Proof. From optimality we have for all $k_n \in \mathcal{N}$:

$$0 \leq \gamma\delta^{k_n} \leq f^{k_n-1}(w^{k_n-1}) - f^{k_n-1}(w^{k_n}) = f^{k_n-1}(w^{k_{n-1}}) - f^{k_n-1}(w^{k_n}).$$

The second inequality follows from $\gamma \in (0,1)$ and the equality follows by noting that $w^{k_n-1} = w^{k_{n-1}}$. The average improvement over M such iterations satisfies:

$$\frac{\gamma}{M}\sum_{m=1}^{M}\delta^{k_m} \leq \underbrace{\frac{1}{M}(f^{k_1-1}(w^{k_0}) - f^{k_M-1}(w^{k_M}))}_{(a)} + \underbrace{\frac{1}{M}\sum_{m=1}^{M}(f^{k_{(m+1)}-1}(w^{k_m}) - f^{k_m-1}(w^{k_m}))}_{(b)}$$

Under assumptions (A1) and (A3), part (a) converges to zero as $M \to \infty$. The term (b) also tends to zero, almost certainly, due to the uniform convergence of the sequence f^k. Hence, we deduce:

$$0 \leq \lim_{k_m \in \mathsf{N}} \inf \delta^{k_m} \leq \lim_{M\to\infty} \frac{1}{M}\sum_{m=1}^{M}\delta^{k_m} = 0.$$

This leads to the conclusion that there exists a sequence, denoted as $\mathcal{K} \subseteq \mathsf{N}$, for which $\{\delta^k\}_{\mathcal{K}} \to 0$. □

While the above result establishes the existence of a subsequence, one can identify such a subsequence computationally. Notice from the above proof that we can construct an index set $\mathcal{K}^*$ using iterations k_m where $\delta^{k_m} \leq \frac{1}{M}\sum_{m=1}^{M}\delta^{k_m}$. We can establish that such an index set is an infinite set (as required in the proof), and every accumulation point of $\{\delta^k\}_{\mathcal{K}^*}$ is zero. In the following theorem, we use Lemma 3.1 and Lemma 3.2 to prove the convergence of $\{w^k\}$ to the optimal solution of (1).

Theorem 3.3. *Let $\{w^k\}_{k=1}^{\infty}$ denote the sequence of incumbent solutions. There exists a subsequence $\{w^k\}_{\mathcal{K}}$ for which every accumulation point is optimal to* (1).

Employing exact procedures to generate minorants has several advantages. From a computational point of view, it offers a direct means to verify the optimality of a given iterate. In the exact bundle method, we track $f(w^{k-1}) - \theta^k$ and terminate when this difference falls below a desired threshold. Both the quantities used to verify the termination criterion are readily available within the algorithm. We refer the reader to [27] for details. When inexact procedures are employed in proximal bundle methods to generate minorants, note that (17) and Lemma 3.2 ensure that there exists a subsequence of iterations, indexed by $\mathcal{K}$ such that $f^{k-1}(w^{k-1}) - \theta^k \to 0$ over $\mathcal{K}$. Although $f^{k-1}(w^{k-1}) - \theta^k$ is a suitable termination metric, stopping the algorithm when this quantity is lower than a threshold may be premature. While a thorough investigation of termination criteria is outside the scope of this paper, we identify a couple of criteria that can be employed when the partition-based inexact cut-generating procedure is used. Firstly, we should expect the collection of partitions to stabilize. Secondly, the partitions identified for any observation, either by solving the subproblem or through the argmax procedure, must remain consistently the same. Designing termination criteria for when the dual-based procedure is employed is more challenging. Nevertheless, in our computational experiments, we run the algorithms for a minimum number of iterations and terminate when the function values at successive candidate solutions differ by, at most, a certain threshold.

3.4 Implementational Details

In this section, we explore the specific procedures involved in implementing the partition-based method. A comprehensive understanding of the implementation details is essential, as it plays a pivotal role in influencing the overall performance of the algorithm, particularly in result storage and executing efficient calculations. Initially, let's delve into the manner in which we structure and store information for use in iterations so that we don't have to retrieve it each time a new iteration occurs. First, for a given observation ω indexed as i in Ω, we decompose the random right-hand sides $\xi(\omega)$ and $\bar{y}(\omega)$ into two parts. The first part is the mean of the random variables that are stored in double vectors bBar and yBar, and the deviation from the mean stored as sparse vectors in `omega-b[i]` and `omega-ybar[i]`. Thus, we have:

$$\xi(\omega) = \texttt{bBar + omega\_b[i]}$$
$$\bar{y}(\omega) = \texttt{yBar + omega\_ybar[i]}.$$

This decomposition enables us to capitalize on the computational efficiency facilitated by sparsity. It allows us to identify and streamline calculations that are consistent across all observations, distinguishing them from those that vary. We exploit the property obtained in equation (10), and for a partition p, we calculate and denote matrix $[W;T]$ as `[Wp; Tp]`. Here, we utilize the solution derived from the mean-value problem as an initial reference for the algorithm. Employing this reference allows us to circumvent redundant calculations throughout the algorithm's execution. We commence the computation as follows to establish a reference solution associated with that specific partition:

$$\texttt{Lambda[p] = [Wp;Tp]*Xmean.} \tag{18}$$

To minimize unnecessary memory usage, it's important to highlight that, for the initialization of the lambda structure, we allocate memory for a pointer to double vectors for the maximum number of potential partitions that could be discovered. Whenever we discover a new partition, we assign memory to one of its elements based on the size of the first-stage decision variable. The computation in equation (18) is performed singularly upon discovering a new partition. Next, we elucidate the procedure for computing solutions for the second-stage problem for a given candidate solution x^k and an observation ω. This involves carrying out the subsequent calculations to obtain the second part of the solution for the recently identified partition p across all observations ω:

$$\texttt{Delta[p][i] = [WP;TP]*omega\_b[i].} \tag{19}$$

Calculations in (19) take advantage of the sparsity of `omega_b[i]` in the multiplication. To efficiently assign memory, we initialize the Delta variable, which is a pointer to pointers to pointers of doubles. We allocate memory for the maximum potential partitions, but we do not assign memory for the interior pointers until a new observation is discovered. Thirdly, the other component of the solution depends on x^k. This part is obtained and stored at the beginning of each iteration as follows:

$$\texttt{DeltaX[p] = [WP;TP]*C*(x\_k-Xmean).} \tag{20}$$

Thus, for x^k and ω, to obtain the solution of the second-stage problem associated with partition p, we have:

$$\texttt{sol = Lambda[p] + Delta[p][i] - DeltaX[p].} \tag{21}$$

Thus, `sol` is decomposed into three parts: the first part is independent of ω and x^k, the second part is independent of x^k, and the third part is independent of ω. To compute the cut coefficients α and β in equation (14), we once again streamline the calculations using the same strategy employed for calculating the solution. Since alpha and beta are functions of `sol`, replacing these three parts in (14) yields a decomposed version of α as well. Here, the first part, which is independent of ω and x^k is calculated as follows:

```
Sigma_alpha[p] = -1/2*Lambda[p]_y*Q*Lambda[p]_y +
bBar*Lambda[p]_lambda + ylBar*Lambda[p]_nu - yuBar*Lambda[p]_mu
```

Now, let's explore how to compute the variable part of α. The second part is independent of x^k and is calculated and stored once for every observation. This section is computed as follows:

```
Delta_alpha[p][i] = -1/2*[2.Lambda[p]_y*Q*Delta[p][i]_y +
Delta[p][i]_y*Q*Delta[p][i]_y] + bBar*Delta[p][i]_lambda +
omega_b[i]*Lambda[p]_lambda + omega_b[i]*Delta[p][i]_lambda +
ylBar*Delta[p][i]_nu - (yBar*Delta[p][i]_mu +
omega_ybar[i]*Lambda[p]_mu + omega_ybar[i]*Delta[p][i]_mu)
```

Finally, the subsequent section represents the third part of α that needs to be computed in each iteration for all observations:

```
AlphaX[p][i] = -1/2*[2*Lambda[p]_y*Q*DeltaX[p]_y +
2*Delta[p][i]_y*Q*DeltaX[p][i]_y + DeltaX[p]_y*Q*DeltaX[p]_y] +
bBar*DeltaX[p]_lambda + omega-b[i]*DeltaX[p]_lambda  +
ylBar*DeltaX[p]_nu - (omega_ybar[i]*DeltaX[p]_mu + yBar* DeltaX[p]_mu),
```

Same procedure is conducted for the calculation slop β as well:

```
sigma-beta[p] = Lambda[p]_lambda*C.
```

In addition, the second and third parts that are varying are calculated as follows:

```
Delta_beta[p] = Delta[p][i]_lambda*C.
```

```
BethaX[p] = DeltaX[p]_lambda*C.
```

4 Numerical Results

In this section, we present the results from the numerical experiments conducted on both the exact and inexact variants of the proximal bundle algorithm presented in this paper. We design our experiments to demonstrate the performance of the inexact compared to the exact proximal bundle algorithm. For the inexact algorithms, we explore their behavior when different fractions of subproblems are solved in every iteration. We also

investigate the performance of these algorithms under different variances of the underlying uncertainty. Before we report the results from these experiments, we share the experimental setup.

We conducted these experiments on instances of two problems encountered in power system planning and operations. The first problem, referred to as `PGP2`, is adapted from its original form and is based on data obtained from [14]. `PGP2` is a two-stage stochastic problem that optimizes power generation capacity increments through reserve facility installation. The goal is to minimize the total costs, including both installation and expected operating costs. This problem involves six variables and two constraints in the first stage and 23 variables and seven constraints in the second-stage problem. The second problem is a modified IEEE-30 test system [4], a standard test system widely employed in power systems analysis. While the first problem is a power systems planning problem, `SODA30` concerns an operations problem known as economic dispatch. Economic dispatch refers to optimizing power generation and distribution in a power system to meet demand efficiently. The system features 30 buses, 41 lines, six generators, and 21 loads. The renewable generator supply is considered random in our experiments. We refer to this problem as `SODA30`, and it has a first stage involving 72 and 98 constraints, while the second stage comprises 80 constraints and 132 variables.

For the computational experiments, we utilized a system with Ubuntu 22.04.3 LTS operating system equipped with an Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz, featuring a quad-core architecture with a maximum clock speed of 3.9 GHz and supported by 7.5 GB of RAM. All three algorithms were implemented using C programming language. The optimization procedures were carried out using the Gurobi Optimizer (version 9.5), and the linear algebra routines were performed using LAPACK (Linear Algebra PACKage). Our implementation serves as a general-purpose solver for 2-SQP as it accepts input in SMPS file format [12]. Furthermore, we developed custom data structures that take advantage of the sparsity of the problems. The instances of the two problems used in our experiments and the algorithm source codes are available at the authors' GitHub repository (`https://github.com/SMU-SODA/stochasticQP`).

We conduct 30 replications of all our experiments, each with randomly and independently generated instances of the above-mentioned problems. In presenting the results of the experiments, we report the metrics averaged over the 30 replications. We also report the standard deviations of the obtained metrics that are provided in parentheses adjacent to the corresponding results. The results for the partition-based inexact bundle algorithm utilize the aggressive-dual approach for selecting subsets in PDAS. We present a comparison with other approaches in section 4.4.

4.1 Efficiency comparison of exact and inexact algorithms

The first experiment elucidates the comparative performance of the exact and inexact proximal bundle algorithms in different metrics. The results are presented in Table 1. The table shows the number of observations in the problem instance ($N = |\Omega|$), the total number of iterations before the stopping condition is satisfied (Iterations), the objective function value associated with the last incumbent solution (Obj. value), the total computational time (Time), which is reported in seconds, and the relative difference between

the objective function value reported by the exact and inexact algorithms (Δ). The optimal value derived from the exact algorithm closely approximates the true optimal value. Notice that as the value of N increases, the estimated objective function value tends to rise while the associated standard deviation tends to decrease. This behavior is consistent with the sample average approximation theory (see [28]). Although this pattern is evident in the results for PGP2, it appears somewhat more irregular in the case of SODA30. In this experiment, we solved only 10% of subproblems when inexact procedures were employed. For both problems, the inexact methods achieve objective function values that are comparable to those obtained when the exact method is used. This is evident from the values of Δ in the table, which are less than 0.0145 and 0.0076 for PGP2 and SODA30, respectively.

We noticed that, with comparable objective function values, the partition-based inexact bundle algorithm outperforms the exact algorithm in terms of the total computational time. For instance, when considering $N = 500$, the exact algorithm requires 11.20 seconds for the PGP2 instance and 37.71 seconds for the SODA30 instance. In contrast, the partition-based algorithm solves these instances in 5.53 seconds for PGP2and 10.36 seconds for SODA30. Since the number of iterations is comparable in the two algorithms, this observation is attributed to the per-iteration computational effort in solving subproblems for all observations when compared to using the PDAS method and argmax step. We emphasize the role of the carefully designed data structures and subroutines used in implementing the PDAS method and argmax step in achieving this computational performance improvement.

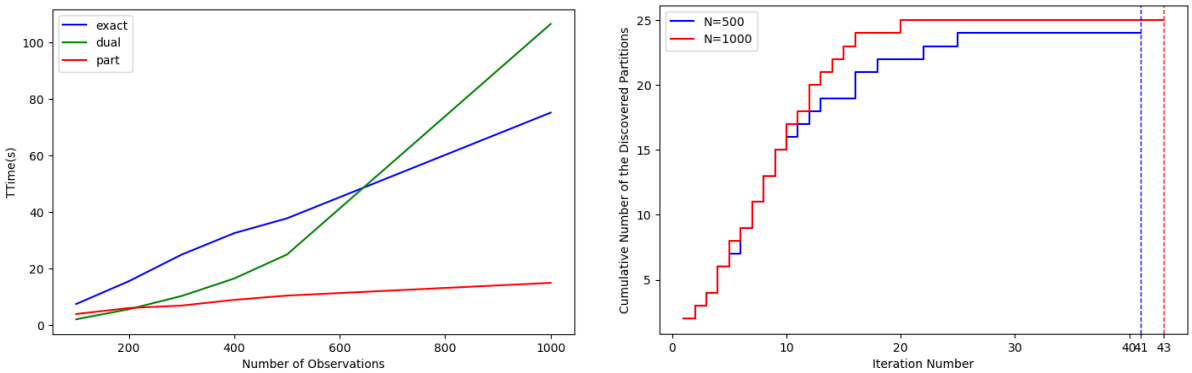
The dual-based inexact bundle algorithm exhibits the best performance in terms of the total computational time in PGP2. In SODA30, while it has the best performance when $N \leq 200$, the dual-based algorithm shows the poorest performance when N exceeds 200 (see Figure 1a). This decline in performance is attributed to the substantial expenses associated with the resource-intensive argmax step in the dual-based algorithm, which necessitates an exhaustive search across all acquired dual solutions. To substantiate this point, we assessed the sizes of the sets Π and $\mathcal{P}$ in the largest instances of PGP2 and SODA30 (with 600 and 1000 observations, respectively). For the dual-based algorithm, the sizes were 2665 and 4075 for PGP2 and SODA30, respectively. In contrast, for the partition-based algorithm, the sizes were 270 and 28. This disparity elucidates why the performance of the dual-based algorithm is significantly poorer for SODA30. As the problem size grows, we expect the set Π to enlarge as observed when transitioning from PGP2 to SODA30; thereby increasing the computational cost of the argmax step. However, it is still possible for the set $\mathcal{P}$ to remain within a reasonable size. While it is impossible to predict the number of partitions one may encounter, based on our empirical observation, we expect the partition-based algorithm to outperform the dual-based algorithm in large-scale settings.

Figure 1b illustrates the progress of the partition-based algorithm (number of iterations along the horizontal axis) in terms of discovering new partitions. The vertical axis shows the cumulative number of partitions discovered. The figure illustrates the progress in two instances of SODA30 with 500 and 1000 observations, respectively. In the case of 500 observations, the total number of iterations amounted to 41. Notably, the latest iteration wherein a new partition emerged from solving second-stage problems was iteration 25, resulting in a total of 24 identified partitions. In the case of 1000 observations, the

Table 1: Numerical results for PGP2 and SODA30.

Algorithm	N	Iterations	Obj. value	Time(s)	Δ
PGP2					
Exact	100	43.16 (2.19)	556.97 (10.18)	1.84 (0.21)	-
	200	43.90 (3.43)	558.98 (8.20)	4.22 (0.42)	-
	300	43.86 (3.00)	559.04 (5.43)	6.79 (0.55)	-
	400	44.9 (3.12)	560.64 (5.12)	9.31 (0.76)	-
	500	44.66 (3.31)	560.44 (4.53)	11.20 (0.80)	-
	600	43.70 (2.96)	561.20 (3.74)	13.21 (1.02)	-
Dual-based	100	50.33 (7.51)	555.97 (10.37)	0.33 (0.06)	0.0018
	200	47.23 (7.08)	558.64 (8.16)	0.78 (0.19)	0.0006
	300	45.63 (4.15)	558.64 (5.38)	1.41 (0.19)	0.0008
	400	45.80 (3.95)	560.36 (5.21)	2.33 (0.30)	0.0005
	500	44.40 (4.03)	560.14 (4.55)	3.33 (0.53)	0.0005
	600	44.2 (3.73)	560.95 (3.81)	4.73 (0.73)	0.0006
Partition-based	100	50.20 (7.90)	548.86 (25.12)	0.74 (0.15)	0.0145
	200	47.16 (5.29)	558.32 (8.57)	1.46 (0.21)	0.0014
	300	44.93 (3.35)	558.48 (4.88)	2.45 (0.96)	0.0010
	400	47.76 (5.06)	560.42 (5.28)	4.70 (0.80)	0.0005
	500	45.20 (4.09)	560.32 (4.47)	5.53 (1.00)	0.0003
	600	45.86 (4.40)	561.01 (3.71)	7.77 (1.21)	0.0005
SODA30					
Exact	100	42.53 (2.72)	-264521 (141)	7.37 (0.51)	-
	200	43.40 (2.85)	-264529 (79)	15.46 (1.18)	-
	300	43.30 (2.64)	-264504 (89)	24.87 (1.75)	-
	400	43.86 (2.66)	-264490 (73)	32.48 (2.13)	-
	500	42.56 (2.60)	-264487 (71)	37.71 (2.56)	-
	1000	43.20 (2.24)	-264492 (49)	75.11 (4.30)	-
Dual-based	100	40.20 (0.48)	-264819 (1718)	1.97 (0.25)	0.0048
	200	40.30 (0.79)	-264937 (1337)	5.53 (0.41)	0.0041
	300	40.73 (2.27)	-265016 (1686)	10.21 (1.22)	0.0043
	400	40.43 (1.04)	-265430 (1814)	16.42 (0.99)	0.0051
	500	40.73 (1.25)	-264775 (1297)	24.92 (1.90)	0.0036
	1000	40.40 (0.85)	-265202 (3332)	106.51 (6.83)	0.0076
Partition-based	100	44.43 (3.23)	-264520 (141)	3.83 (1.04)	$<$ 10e-4
	200	44.06 (3.62)	-264530 (78)	5.96 (2.31)	$<$ 10e-4
	300	43.06 (3.23)	-264505 (90)	6.83 (0.84)	$<$ 10e-4
	400	43.30 (2.43)	-264491 (73)	8.85 (1.30)	$<$ 10e-4
	500	43.26 (2.55)	-264487 (70)	10.36 (0.71)	$<$ 10e-4
	1000	42.9 (2.78)	-264493 (49)	14.87 (1.50)	$<$ 10e-4

last new partition was discovered in iteration 20, and the algorithm ran for 43 iterations. Recall that the convergence analysis of the partition-based algorithm relies on encountering the same partition after a finite number of iterations for any observation. In other words, we expect that all relevant partitions are discovered after a certain number of iterations. The figure provides empirical evidence of this fact.



(a) Comparison of Computational Time (b) Partition Discovery Pattern

Figure 1: Comparison of Efficiency of Proximal Bundle methods

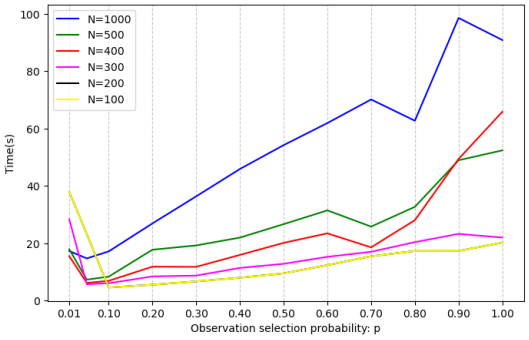


Figure 2: Impact of Sampling Fraction on Partition-based Method (Instances of SODA30)

4.2 Impact of Sampling Fraction on Partition-based

As previously discussed in Remark 3.1, the asymptotic performance of the algorithm is achieved under different subset selection criteria. Since the per iteration computational cost of inexact algorithms is proportional to the number of subproblems solved (size of $|\Omega^k|$), our next experiment assesses the impact of the selection strategy. We report the results only for the partition-based algorithm. One criterion that fulfills the necessary conditions for the selection is to sample each observation with a constant nonzero probability, denoted as p, in every iteration. We examined the impact of different p values on a crucial performance metric, the computational time before termination. These results are presented in Figure 2 for instances of SODA30 with different numbers of observations. The figure shows the computational time (vertical axis) for different values of p between 0.01 and 1 along the horizontal axis. Notice that $p = 1$ corresponds to the exact method.

The figure demonstrates that with increasing p values, the computational time decreases, reaching a minimum value at either $p = 0.05$ or $p = 0.10$ for all instances. However, once p values exceed this point, the computational time increases. The higher computational cost for very small p values is attributed to weaker minorants that prolong the algorithm's termination. Similarly, for large p values, the increased computational cost is because of the need to solve a larger number of second-stage QP problems. Nevertheless, these results demonstrate that we can obtain high-quality solutions even by solving a small fraction of subproblems.

4.3 Effect of Variance on the Number of Discovered Partitions

In this experiment, we investigate how the variance of random variables (right-hand sides) influences the performance of the partition-based algorithm. Specifically, we monitor the number of identified partitions. We report the results for the `SODA30` instance with 1000 observations in Table 2. We vary the variance of two random variables, both with mean one, between 0.01 to 4. We applied the partition-based algorithm with a sample fraction of 0.1 to solve these problem instances.

The table shows the total computational time and the number of discovered partitions for different variance values. The results show that as the variability on the right-hand side amplifies, it leads to heightened variability in the second-stage subproblems. Consequently, in line with expectations, there is an escalation in the number of partitions identified. This increases the computational effort in PDAS and argmax step which is aggregated in the computational time mentioned in the table.

4.4 Comparison of different versions of implementation of PDAS

In our last experiment, we compare the different subset selection approaches within PDAS, namely Single-Swap, Aggressive-Dual, Aggressive-Primal, and Aggressive-Primal-Dual. For this experiment, we use a deterministic instance of `SODA30` (a QP problem with linear constraints). Beginning with the same random partition as input, we run the PDAS method until it discovers an optimal solution. Additionally, we note the results when the first feasible dual solution is discovered. These results are reported in Table 3. In this table, we report the range of relative gaps (it is a range across 30 replications) between the optimal objective function value and the objective function at the solutions reported from PDAS. We observed cycles with all four subset selection approaches while looking for the optimal solution. In the PDAS method, a cycle occurs if we encounter the same partition as one obtained in a previous iteration. We document the number of instances where the algorithm encounters cycles (# Cycles). The average computational time and the range of relative gaps are computed only for instances where we did not encounter cycles.

Analysis of results reported in Table 3 reveals that the runtimes of the four algorithms are closely comparable. The Single-Swap approach provides the best objective function values for all instances where we did not encounter cycles, with the smallest relative gap of 0.04. This indicates the possibility of achieving a high-quality objective function value from the Single-Swap approach compared to the other approaches. Single-Swap also exhibits the lowest occurrence of cycles when searching for the optimal solution. It is noteworthy that the Aggressive-Dual approach consistently avoids encountering cycles

Table 2: Effect of Variance on Discovered Partitions

Variance	0.01	0.10	0.50	1.00	1.50	2.00	2.50	3.00	4.00
# Partitions	3.19	7.83	8.34	8.30	8.99	8.18	11.47	11.06	11.73
Time(s)	7.5	15.4	24.7	25.2	26.9	28.0	30.7	31.7	33.6

Table 3: Numerical results for PDAS

Approach	Dual feasibility			Optimality		
	Time(s)	Gap	# Cycles	Time(s)	Gap	# Cycles
Single-Swap	5.77×10^{-6}	[0.04,294.27]	3	4.90×10^{-6}	0.00	10
Aggressive-Dual	6.00×10^{-6}	[145.29,295.86]	0	5.08×10^{-6}	0.00	17
Aggressive-Primal	6.27×10^{-6}	[246.73 ,295.86]	12	-	-	30
Aggressive-Primal-Dual	5.95×10^{-6}	[26.76,295.86]	8	-	-	30

while searching for a feasible dual solution. This observation aligns with our expectations, as the Aggressive-Dual approach, in its pursuit of feasible duality, strives to push the solution into the dual feasible space without considering primal feasibility. This approach also showed better performance with respect to the number of times it faced cycles while searching for optimal solutions. Both Single-Swap and Aggressive-Dual approaches resulted in a 0.00 optimality gap in all instances they could solve. In contrast, we encountered cycles when solved to optimality in all instances with both Aggressive-Primal and Aggressive-Primal-Dual. We must guarantee that the PDAS method provides a feasible dual solution when employed in the partition-based proximal bundle algorithm. The results from this experiment provide empirical justification for using the Aggressive-Dual approach in implementing the PDAS method within the partition-based proximal bundle algorithm.

5 Conclusions

In this paper, we initiated our exploration by examining the impact of varying the right-hand side on the optimal partition of a QP problem. To compute the solution of the associated QP problem for a given partition, we introduced a linear system derived from the reduced KKT system. Subsequently, we presented four distinct implementations of the PDAS method, specifically tailored for projecting an infeasible dual solution of the QP problem into the feasible region. We proposed two variations of the inexact proximal bundle algorithm — namely, dual-based and partition-based, customized for 2-SQP problems. A key feature of these algorithms is their capability to guarantee global convergence while accommodating imprecision in solving second-stage problems during the cut-generation process within the proximal bundle algorithm. The numerical experiments that compare the inexact algorithms with the exact proximal bundle algorithm demonstrate that the partition-based algorithm surpasses the exact algorithm in terms of total computational time while delivering a similar quality solution. The dual-based algorithm exhibits less favorable performance, particularly when the sample size increases. Development of reliable stopping rules, convergence-rate analysis, and extensions to multistage SQP are fruitful future directions that we intend to pursue.

A Omitted Proofs

Proof of Proposition 2.1. In the proof of this proposition, we will refer to the problem (2) with right-hand side and upper bound $(\rho, \bar{y})$ as the original problem and the one with right-hand side and upper bound $(\rho', \bar{y}')$ as the perturbed problem. We define $\Delta\rho = \rho' - \rho$ and $\Delta\bar{y} = \bar{y}' - \bar{y}$. Consider the problem with the right-hand side and upper bound $(\rho + \Delta\rho, \bar{y} + \Delta\bar{y})$. By using the partition $(\mathsf{L}, \mathsf{U}, \mathsf{I})$, we can construct a feasible solution for this perturbed problem. Specifically, the primal variables that belong to the active set can be obtained directly from the definition of the active set, while the values of the dual variables corresponding to inactive primal variables can be determined via complementarity slackness. That is, $y'_{\mathsf{U}} = \bar{y}_{\mathsf{U}} + \Delta\bar{y}_{\mathsf{U}}$; $y'_{\mathsf{L}} = \underline{y}_{\mathsf{L}}$; $\mu'_{\mathsf{L}} = 0$; $\nu'_{\mathsf{U}} = 0$; $\mu'_{\mathsf{I}} = 0$ and $\nu'_{\mathsf{I}} = 0$. Using these, the system of equations (7) for the perturbed problem can be decomposed into a reduced KKT system as:

$$\begin{bmatrix} Q_{\mathsf{II}} & D_{\mathsf{MI}}^T \\ D_{\mathsf{MI}} & 0 \end{bmatrix} \begin{bmatrix} y'_{\mathsf{I}} \\ \lambda' \end{bmatrix} = - \begin{bmatrix} Q_{\mathsf{IL}} \\ D_{\mathsf{ML}} \end{bmatrix} \underline{y}_{\mathsf{L}} - \begin{bmatrix} Q_{\mathsf{IU}} \\ D_{\mathsf{MU}} \end{bmatrix} \bar{y}'_{\mathsf{U}} - \begin{bmatrix} d_{\mathsf{I}} \\ -\rho' \end{bmatrix}, \tag{22}$$

$$\begin{bmatrix} \nu'_{\mathsf{L}} \\ \mu'_{\mathsf{U}} \end{bmatrix} = \begin{bmatrix} Q_{\mathsf{LL}} \\ -Q_{\mathsf{UL}} \end{bmatrix} \underline{y}_{\mathsf{L}} + \begin{bmatrix} Q_{\mathsf{LU}} \\ -Q_{\mathsf{UU}} \end{bmatrix} \bar{y}'_{\mathsf{U}} + \begin{bmatrix} Q_{\mathsf{LI}} & D_{\mathsf{ML}}^T \\ -Q_{\mathsf{UI}} & -D_{\mathsf{MU}}^T \end{bmatrix} \begin{bmatrix} y'_{\mathsf{I}} \\ \lambda' \end{bmatrix} + \begin{bmatrix} d_{\mathsf{L}} \\ -d_{\mathsf{U}} \end{bmatrix}. \tag{23}$$

Note that the first system of equations can be used to solve for y'_{I} and λ'. Using this solution, we can solve the second system of equations for ν'_{L} and μ'_{U}. Studying the two systems of equations separately allows us to isolate the effect of changes to the right-hand side and upper-bound elements on the two sets of variables. The equation (22) can be simplified using the matrices M and W given in (8). This results in an expression that represents the primal and dual solutions of the perturbed problem as a function of the corresponding values for the original problem and the perturbations $\Delta\rho$ and $\Delta\bar{y}$. As a result, after performing the necessary calculations, we obtain the following equation:

$$\begin{bmatrix} y'_{\mathsf{I}} \\ \lambda' \end{bmatrix} = \underbrace{M\left(- \begin{bmatrix} Q_{\mathsf{IL}} \\ D_{\mathsf{ML}} \end{bmatrix} \underline{y}_{\mathsf{L}} - \begin{bmatrix} Q_{\mathsf{IU}} \\ D_{\mathsf{MU}} \end{bmatrix} \bar{y}_{\mathsf{U}} - \begin{bmatrix} d_{\mathsf{I}} \\ -\rho \end{bmatrix} \right)}_{=[y_{\mathsf{I}}, \lambda]^\top} + \underbrace{W \begin{bmatrix} \Delta\bar{y}_{\mathsf{U}} \\ \Delta\rho \end{bmatrix}}_{:=[\Delta y_{\mathsf{I}}, \Delta\lambda]}. \tag{24}$$

Substituting the expressions we obtained for y'_{I} and λ' in (23), we obtain an analogous expression for the dual variables associated with the bounds of the perturbed problem in terms of their counterparts in the original problem and the perturbation. Thus, we have:

$$\begin{bmatrix} \nu'_{\mathsf{L}} \\ \mu'_{\mathsf{U}} \end{bmatrix} = \underbrace{- \begin{bmatrix} -Q_{\mathsf{LL}} & -Q_{\mathsf{LU}} \\ Q_{\mathsf{UL}} & Q_{\mathsf{UU}} \end{bmatrix} \begin{bmatrix} \underline{y}_{\mathsf{L}} \\ \bar{y}_{\mathsf{U}} \end{bmatrix} - \begin{bmatrix} -Q_{\mathsf{LI}} & -D_{\mathsf{ML}}^T \\ Q_{\mathsf{UI}} & D_{\mathsf{MU}}^T \end{bmatrix} \begin{bmatrix} y_{\mathsf{I}} \\ \lambda \end{bmatrix} - \begin{bmatrix} -d_{\mathsf{L}} \\ d_{\mathsf{U}} \end{bmatrix}}_{=[\nu_{\mathsf{L}}, \mu_{\mathsf{U}}]^\top} + \underbrace{T \begin{bmatrix} \Delta\bar{y}_{\mathsf{U}} \\ \Delta\rho \end{bmatrix}}_{:=[-\Delta\nu_{\mathsf{L}}, \Delta\mu_{\mathsf{U}}]^\top},$$

where T is given in (9). Therefore, the solution to the perturbed problem can be decomposed into a solution of the original problem and a displacement term. The solutions of the perturbed problem are dual feasible if

$$\begin{bmatrix} \nu_{\mathsf{L}} \\ \mu_{\mathsf{U}} \end{bmatrix} + T \begin{bmatrix} \Delta\bar{y}_{\mathsf{U}} \\ \Delta\rho \end{bmatrix} \geq 0. \tag{25}$$

Notice that since ν_{L} and μ_{U} are optimal for the original problem, (25) is trivially satisfied for $[\Delta\bar{y}, \Delta\rho]^\top = 0$. Using the above observation, we can construct a polyhedron S_1 as $S_1 = \{(\rho', \bar{y}') \mid [\Delta\rho, \Delta\bar{y}]^\top \text{ satisfies (25)}\}$. This completes the proof of the first part. If $(\rho', \bar{y}') \in S_1$ yield $(\Delta\rho, \Delta\bar{y})$ that further satisfy the primal feasibility condition

$$\underline{y}_{\mathsf{I}} \leq y_{\mathsf{I}} + W_{\mathsf{I}(\mathsf{U}+\mathsf{M})} \begin{bmatrix} \Delta\bar{y}_{\mathsf{U}} \\ \Delta\rho \end{bmatrix} \leq \bar{y}_{\mathsf{I}} + \Delta\bar{y}_{\mathsf{I}}, \tag{26}$$

then the resulting solution for the perturbed problem satisfies first-order optimality conditions. Notice that the conditions in (26) are affine in $(\Delta\rho, \Delta\bar{y})$. Therefore, for the elements of the polyhedron $S_2 = \{(\rho', \bar{y}') \mid [\Delta\rho, \Delta\bar{y}]^\top \text{ satisfies (25) and (26)}\}$, the current partition $(\mathsf{L}.\mathsf{U}, \mathsf{I})$ remains optimal. This completes the proof. □

Proof of Lemma 3.1. Consider the function $f^k(x)$, composed of two distinct parts. The first part is a polynomial function given by $\frac{1}{2}x^\top Px + c^\top x$. Due to the inherent continuity of polynomial functions, it follows that $\frac{1}{2}(x^k)^\top Px^k + c^\top x^k \to \frac{1}{2}\hat{x}^\top P\hat{x} + c^\top\hat{x}$, whenever the sequence $\{x^k\} \to \hat{x}$. We next establish the convergence of the second part of $f^k(x)$, i.e., $\max_{j\in J^k}(\alpha^j + (\beta^j)^\top x^k) \to \mathbb{E}[h(\hat{x}, \tilde{\omega})]$. In this regard, from Proposition 2.21 of [28], note that the $h(x, \omega)$ is a continuous function. Now, let us define $\mathcal{J}_\omega$ as the sequence of iterations when the subproblem associated with ω is solved exactly. We consider the rest of the proof for the dual-based and partition-based inexact bundle algorithms distinctly.

Dual-based algorithm: In the dual-based algorithm, we have $\Pi^k \subseteq \Pi^{k+1} \subseteq \ldots \subseteq \Pi$, and the argmax step utilizes this collection of dual solutions. Therefore, if we define $h^k(x, \omega) = \max\left\{\frac{1}{2}y^\top Qy + (\xi(\omega) - Cx)^\top\lambda + \underline{y}^T\nu - \bar{y}(\omega)^\top\mu \mid \pi \in \Pi^k\right\}$, we observe the following relationship for all $\omega \in \Omega$:

$$h^k(x, \omega) \leq h^{k+1}(x, \omega) \leq \ldots \leq h(x, \omega). \tag{27}$$

Since $\mathcal{X}$ is compact, the monotonic sequence of continuous functions above converges uniformly. Following the continuity of h and convergence of $\{x^k\}_{\mathcal{K}}$, we have for every $\epsilon > 0$ there exists $K < \infty$, such that for $j \in \mathcal{J}_\omega \cap \mathcal{K}$ and $j > K$, we have:

$$|h(\hat{x}, \omega) - h^j(x^j, \omega)| = |h(\hat{x}, \omega) - h(x^j, \omega)| < \frac{\epsilon}{3}. \tag{28}$$

The equality is because we solved the subproblem exactly. Since h^j is a continuous function, there exists $K' < \infty$ such that:

$$|h^j(x^j, \omega) - h^j(x^k, \omega)| < \frac{\epsilon}{3},$$

for all $k, j \in \mathcal{K}$ and $k, j > K'$. From uniform convergence of $\{h^k\}$,there exists $K'' < \infty$ such that:

$$|h^j(x^k, \omega) - h^k(x^k, \omega)| < \frac{\epsilon}{3}, \tag{29}$$

for every $k, j > K''$. Thus, for $j \in \mathcal{J}_\omega$ and $k \in \mathcal{K}$ such that $k > j > \max(K, K', K'')$, we have

$$\begin{aligned} |h(\hat{x}, \omega) - h^k(x^k, \omega)| \leq\ & |h(\hat{x}, \omega) - h^j(x^j, \omega)| \\ & + |h^j(x^j, \omega) - h^j(x^k, \omega)| + |h^j(x^k, \omega) - h^k(x^k, \omega)| < \epsilon. \end{aligned}$$

The sampling procedure guarantees the existence of a j_ω that satisfies the above requirement for all $\omega \in \Omega$. Therefore, $\left| \sum_{\omega\in\Omega} p(\omega)(h(\hat{x},\omega) - \left(\alpha^k(\omega) + \beta^k(\omega)x^k\right) \right| \leq \epsilon$, for $k > \max_{\omega\in\Omega}\{j_\omega\}$. We have used $h^k(x^k,\omega) = \alpha^k(\omega) + \beta^k(\omega)x^k$ in the above. The result is thus established.

Partition-based algorithm: Here, we define $h^k(x,\omega) = \max\left\{\frac{1}{2}y^\top Qy + (\xi(\omega) - Cx)^\top\lambda + \underline{y^T}\nu - \bar{y}(\omega)^\top\mu \ |\pi \in \Pi(x,\omega,\mathcal{P}^k)\right\}$. We observe that a similar relationship as (27) holds in the partition-based method since $\mathcal{P}^k \subseteq \mathcal{P}^{k+1} \subseteq \ldots \subseteq \mathcal{P}$ and the argmax procedure uses the collection of partitions. For every $\epsilon > 0$, let us define parameter K as (28).As a consequence of Proposition 2.1; there exists a ball with radius $\delta(\omega)$ centered at $\hat{x}$ such that the optimal partition is the same for $x \in \{\chi| \ ||\chi - \hat{x}|| \leq \delta(\omega)\}$ for every $\omega \in \Omega$. Finally, since x^k converges to $\hat{x}$, there exists a K' such that whenever $k > K'$, we have $||x^k - \hat{x}|| < \delta(\omega)$ for all $\omega \in \Omega$. The sampling procedure ensures that we have a $j_\omega \in \mathcal{J}_\omega$ and $j_\omega > \max(K, K')$. Therefore, we have

$$|h(\hat{x},\omega) - h(x^k,\omega)| = |h(\hat{x},\omega) - h^k(x^k,\omega)| \leq \epsilon. \tag{30}$$

for all $k > j_\omega$. The rest of the proof follows as in the case for the dual-based method. □

Proof of Lemma 3.2. We study two possible cases depending on whether $\mathcal{N}$ is finite or infinite. Here, we present the proof under the case When $\mathcal{N}$ is finite, there exists a $K < \infty$ such that for all $k > K$, we have $f^k(x^k) > f^k(\bar{w}) + \gamma(f^{k-1}(x^k) - f^{k-1}(\bar{w}))$, and $w^{k-1} = \bar{w}$. Since $f^k(\bar{w}) \geq f^{k-1}(\bar{w})$

$$f^k(x^k) > f^{k-1}(x^k) + (1-\gamma)[f^{k-1}(\bar{w}) - f^{k-1}(x^k)].$$

Let $g^k \in \partial f^k(x^k)$. From the subgradient inequality and above inequality, we have:

$$f^k(x) \geq f^k(x^k) + \langle g^k, x - x^k\rangle \geq f^{k-1}(x^k) + (1-\gamma)\delta^k + \langle g^k, x - x^k\rangle. \tag{31a}$$

Since x^k is the minimizer of (12), we have $0 \in \partial f^{k-1}(x^k) + \sigma(x^k - \bar{w})$. Again, from subgradient inequality, we have:

$$f^k(x) \geq f^{k-1}(x) \geq f^{k-1}(x^k) + \sigma\langle(\bar{w} - x^k), (x - x^k)\rangle. \tag{31b}$$

Thus, from (31a) and (31b), we have:

$$\begin{aligned}
f^k(x) &\geq \max\{f^{k-1}(x^k) + (1-\gamma)\delta^k + \langle g^k, x - x^k\rangle, f^{k-1}(x^k) + \sigma\langle(\bar{w} - x^k), (x - x^k)\rangle\} \\
&= f^{k-1}(x^k) + \sigma\langle\bar{w} - x^k, x - x^k\rangle + \max\{(1-\gamma)\delta^k + \langle g^k - \sigma(\bar{w} - x^k), x - x^k\rangle, 0\}. \\
\Rightarrow f^k(x) + \frac{\sigma}{2}||x - \bar{w}||^2 &\geq f^{k-1}(x^k) + \frac{\sigma}{2}||(x^k - \bar{w}) - (x^k - x)||^2 + \\
&\quad \sigma\langle\bar{w} - x^k, x - x^k\rangle + \max\{(1-\gamma)\delta^k + \langle g^k - \sigma(\bar{w} - x^k), x - x^k\rangle, 0\} \\
&= f^{k-1}(x^k) + \frac{\sigma}{2}||x^k - \bar{w}||^2 + \frac{\sigma}{2}||x^k - x||^2 + \\
&\quad \max\{(1-\gamma)\delta^k + \langle g^k - \sigma(\bar{w} - x^k), x - x^k\rangle, 0\}.
\end{aligned}$$

Specifically by setting $x = x^{k+1}$, we have

$$\theta^{k+1} - \theta^k \geq \frac{\sigma}{2}||x^k - x^{k+1}||^2 + \max\{(1-\gamma)\delta^k + \langle g^k - \sigma(\bar{w} - x^k), x^{k+1} - x^k\rangle, 0\}.$$

Therefore, the sequence $\{\theta^k\}$ is nondecreasing and bounded above by $f(\bar{w})$. Therefore, we must have $\theta^{k+1} - \theta^k \to 0$. More generally, noting that $\|g^k\| \leq C$ and defining $\Delta x = ||x - x^k||$, we have

$$\begin{aligned}
\theta^{k+1} - \theta^k \geq & \min_{\Delta x} \Bigg\{ \underbrace{\frac{\sigma(\Delta x)^2}{2} + \max\{(1-\gamma)\delta^k - 2C\Delta x, 0\}}_{:=H(\Delta x)} \Bigg\} \\
\geq & \begin{cases} \frac{\sigma(1-\gamma)^2(\delta^k)^2}{8C^2} & \text{if } (1-\gamma)\delta^k \leq \frac{4C^2}{\sigma} \\ \frac{(1-\gamma)\delta^k}{2} & \text{otherwise,} \end{cases} \\
\geq & \ 0.
\end{aligned}$$

The last inequality follows because $\delta^k \geq 0$. Since $\theta^{k+1} - \theta^k \to 0$, we conclude that $\delta^k \to 0$ when $\mathcal{N}$ is finite. □

Proof. Proof of Theorem 3.3. From Lemma 3.2, we have a sequence $\mathcal{K}$ such that $\{\delta^k\}_{\mathcal{K}} \to 0$. Thus, assumption (A1) and $0 \leq \frac{\sigma}{2}||w^k - w^{k-1}|| \leq \delta^k$ ensure the existence of an accumulation point $\bar{w} \in \mathcal{X}$. From Lemma 3.1, we have $\lim_{k\in\mathcal{K}} f^{k-1}(w^k) = f(\bar{w})$. Combining the two, we have

$$\lim_{k\in\mathcal{K}} \theta^k = \lim_{k\in\mathcal{K}} f^{k-1}(w^k) + \frac{\sigma}{2}||w^k - w^{k-1}|| = f(\bar{w}).$$

Now, if $\bar{w}$ is not optimal. Let $f_\sigma(w)$ denote the Moreau–Yosida regularization of f given by $f_\sigma(w) \coloneqq \min_{x\in\mathcal{X}} \left\{ f(x) + \frac{\sigma}{2}\|x - w\|^2 \right\}$. By Lemma 8 of [27], we have $f_\sigma(w^{k-1}) < f(w^{k-1})$. Since $f^{k-1}(x) \leq f(x)$ for all $x \in \mathcal{X}$, we have $\theta^k \leq f_\sigma(w^{k-1})$. So, we have $f(\bar{w}) = \lim_{k\in\mathcal{K}} \theta^k < \lim_{k\to\infty} f(w^{k-1}) = f(\bar{w})$, which leads to a contradiction. □

References

[1] J. R. Birge and F. Louveaux, *Introduction to stochastic programming*, Springer Science & Business Media, 2011.

[2] J. R. Birge, L. Qi, and X. Chen, *A stochastic Newton method for stochastic quadratic programs with recourse*, Citeseer, 1994.

[3] X. Chen, L. Qi, and R. S. Womersley, *Newton's method for quadratic stochastic programs with recourse*, Journal of Computational and Applied Mathematics, 60 (1995), pp. 29–46.

[4] R. Christie, *Power systems test case archive. 1999*, 2015.

[5] F. E. Curtis and Z. Han, *Globally convergent primal-dual active-set methods with inexact subproblem solves*, SIAM Journal on Optimization, 26 (2016), pp. 2261–2283.

[6] F. E. Curtis, Z. Han, and D. P. Robinson, *A globally convergent primal-dual active-set framework for large-scale convex quadratic optimization*, Computational Optimization and Applications, 60 (2015), pp. 311–341.

[7] W. de Oliveira, C. Sagastizábal, and C. Lemaréchal, *Convex proximal bundle methods in depth: a unified analysis for inexact oracles*, Mathematical Programming, 148 (2014), pp. 241–277.

[8] W. S. Dorn, *Duality in quadratic programming*, Quarterly of applied mathematics, 18 (1960), pp. 155–162.

[9] I. Florescu and C. A. Tudor, *Handbook of probability*, John Wiley & Sons, 2013.

[10] A. Forsgren, P. E. Gill, and E. Wong, *Primal and dual active-set methods for convex quadratic programming*, Mathematical programming, 159 (2016), pp. 469–508.

[11] H. Gangammanavar, Y. Liu, and S. Sen, *Stochastic decomposition for two-stage stochastic linear programs with random cost coefficients*, INFORMS Journal on Computing, 33 (2021), pp. 51–71.

[12] H. I. Gassmann, *The SMPS format for stochastic linear programs*, in Applications of stochastic programming, SIAM, 2005, pp. 9–19.

[13] J. L. Higle and S. Sen, *Finite master programs in regularized stochastic decomposition*, Mathematical Programming, 67 (1994), pp. 143–168.

[14] J. L. Higle and S. Sen, *Stochastic decomposition: a statistical method for large scale stochastic linear programming*, Springer Science & Business Media, 1996.

[15] M. Hintermüller, K. Ito, and K. Kunisch, *The primal-dual active set strategy as a semismooth newton method*, SIAM Journal on Optimization, 13 (2002), pp. 865–888.

[16] K. Ito and K. Kunisch, *The primal-dual active set method for nonlinear optimal control problems with bilateral constraints*, SIAM Journal on Control and Optimization, 43 (2004), pp. 357–376.

[17] K. C. Kiwiel, *A proximal bundle method with approximate subgradient linearizations*, SIAM Journal on optimization, 16 (2006), pp. 1007–1023.

[18] C. Lemaréchal, *Nonsmooth optimization and descent methods*, Tech. Report Research Report RR-78-4, International Institute of Applied Systems Analysis, 1978.

[19] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov, *New variants of bundle methods*, Mathematical programming, 69 (1995), pp. 111–147.

[20] J. Liu and S. Sen, *Asymptotic results of stochastic decomposition for two-stage stochastic quadratic programming*, SIAM Journal on Optimization, 30 (2020), pp. 823–852.

[21] S. Mehrotra and M. G. Ozevin, *Decomposition based interior point methods for two-stage stochastic convex quadratic programs with recourse*, Operations Research, 57 (2009), pp. 964–974.

[22] J. Nocedal and S. Wright, *Numerical optimization*, Springer Science & Business Media, 2006.

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

[24] R. T. Rockafellar and R.-B. Wets, *A lagrangian finite generation technique for solving linear-quadratic problems in stochastic programming*, in Stochastic Programming 84 Part II, Springer, 1986, pp. 63–93.

[25] R. T. Rockafellar and R.-J. Wets, *Scenarios and policy aggregation in optimization under uncertainty*, Mathematics of operations research, 16 (1991), pp. 119–147.

[26] A. Ruszczyński, *A regularized decomposition method for minimizing a sum of polyhedral functions*, Mathematical Programming, 35 (1986), pp. 309–333.

[27] A. Ruszczyński and A. Shapiro, *Stochastic programming models*, Handbooks in operations research and management science, 10 (2003), pp. 1–64.

[28] A. Shapiro, D. Dentcheva, and A. Ruszczynski, *Lectures on stochastic programming: modeling and theory*, SIAM, 2021.

[29] R. R.-J. Wets, *Programming under uncertainty: the equivalent convex program*, SIAM Journal on Applied Mathematics, 14 (1966), pp. 89–105.

[30] R. R.-J. Wets, *Large scale linear programming techniques in stochastic programming*, Springer, Berlin, 1988, ch. 3.

[31] G. Zakeri, A. B. Philpott, and D. M. Ryan, *Inexact cuts in benders decomposition*, SIAM Journal on Optimization, 10 (2000), pp. 643–657.