https://doi.org/10.1016/j.ejor.2019.11.001

# A New Convergent Hybrid Learning Algorithm for Two-Stage Stochastic Programs

Shaorui Zhou[a,*], Hui Zhang[b], Ning Shi[a], Zhou Xu[c], Fan Wang[a]

[a]*School of Business, Sun Yat-sen University, Guangzhou, China*

[b]*School of Public Health, Sun Yat-sen University, Guangzhou, China*

[c]*Department of Logistics & Maritime Studies, The Hong Kong Polytechnic University, Hung Hom, Hong Kong*

## Abstract

This study proposes a new hybrid learning algorithm to approximate the expected recourse function for two-stage stochastic programs. The proposed algorithm, which is called projected stochastic hybrid learning algorithm, is a hybrid of piecewise linear approximation and stochastic subgradient methods. Piecewise linear approximations are updated adaptively by using stochastic subgradient and sample information on the objective function itself. In order to achieve a global optimum, a projection step that implements the stochastic subgradient method is performed to jump out from a local optimum. For general two-stage stochastic programs, we prove the convergence of the algorithm. Furthermore, the algorithm can drop the projection steps for two-stage stochastic programs with network recourse. Therefore, the pure piecewise linear approximation method is convergent when the initial piecewise linear functions are properly constructed. Computational results indicate that the algorithm exhibits rapid convergence.

Keywords: stochastic programming, piecewise linear approximation, network, optimization.

## 1. Introduction

A common challenge in operations research is making a decision in the present such that it minimizes the expectation of costs in the future with uncertainty. For example, several large

[*]Corresponding author
*Email address:* zshaorui@gmail.com (Shaorui Zhou)

shippers, such as Walmart, Amazon, and IKEA, need to decide on the amount of products to ship from plants to warehouses through which they can satisfy the demands of various retailers. Typically, they must make a decision before they know the retail demands. When the retail demands are known, they optimize the shipping schedules between warehouses and retailers. The aforementioned problem can be formulated as a two-stage stochastic program. The decision made in the present (Stage 1) determines the state to begin with while solving the problem in Stage 2. Thus, if the expected cost function (recourse function) for Stage 2 can be computed, then an optimal decision can be made in Stage 1.

## *1.1. Motivation*

A two-stage stochastic program is a practically important problem. Problems of this type arise in several areas in operational management, in which the decision maker must allocate several resources in time and space dimensions prior to realizing events that will affect the outcome of the decisions. For example, in empty container repositioning problems (Cheung and Chen, 1998; Long et al., 2012), shipping companies need to reposition empty containers prior to realizing the demand. In locomotive planning problems (Bouzaiene-Ayari et al., 2016; Cordeau et al., 2000), railroads have to determine the sequence of trains to which each locomotive is assigned before disruptions occur across the network. In job scheduling problems (Kim and Mehrotra, 2015; Restrepo et al., 2017), the manager must determine initial staffing levels and schedules before the actual date of demand realization. In relief distribution problems (Alem et al., 2016; Moreno et al., 2018), humanitarian decision makers must distribute emergency aid to disaster victims when resources are scarce amidst great uncertainty. Most of the above applications are fully sequential problems, and they are modeled as two-stage stochastic programming problems in these papers. Thus, the research into two-stage stochastic optimization is an important foundation. The main difficulty is that in most practical cases of two-stage stochastic programs, the expected recourse function is excessively complex due to uncertainty. In this study, we propose a hybrid learning algorithm called *projected stochastic hybrid learning algorithm* (ProSHLA) to approximate the expected recourse function for two-stage stochastic programs. Furthermore, we theoretically demonstrate the convergence of the algorithm.

Essentially, ProSHLA is a hybrid of piecewise linear approximation and stochastic subgradient methods. The core of ProSHLA involves learning steps that provide information for updating the expected recourse function through a sequence of piecewise linear separable approximations, and the projection steps that guarantee convergence by implementing the stochastic subgradient method. The findings reveal that when initial piecewise linear approximation functions are properly constructed for two-stage stochastic programs with network recourse, the algorithm can drop the projection step without sacrificing convergence. This interesting finding answers the following question that has puzzled researchers for more than a decade: Why does the piecewise linear approximation method work well for stochastic programs with network recourse? Our analytical results provide the first theoretical support for the use of the piecewise linear approximation method in two-stage stochastic programs.

## *1.2. Related literature*

We consider the following two-stage stochastic programming problem:

$$\min_{x \in X} c_0^T x + E_\omega[Q(x, \omega)], \tag{1}$$

subject to

$$Ax = b,$$

$$x \geq 0,$$

where $X \subset \mathbb{R}^n$ denotes a convex compact set and the recourse function $Q(x, \omega)$ denotes the optimal value of the second stage problem.

$$Q(x, \omega) = \min_{y(\omega)} c_1^T y(\omega), \tag{2}$$

subject to

$$W(\omega)y = h(\omega) - T(\omega)x,$$

$$y(\omega) \geq 0(\omega).$$

In the above model, the variables $x$ and $y$ denote the decision variables of Stage 1 and Stage 2 problems, respectively. Parameters $c_0$ and $c_1$ denote the first and second stage vectors of cost coefficients, respectively, and $A$, $W(\omega)$ are constraint matrices.

Several studies have examined stochastic programming models and solution methods. Comprehensive reviews and discussions were performed by Shapiro et al. (2014) and Wallace and Ziemba (2005). Except for a few special cases, the expected recourse function is extremely difficult to evaluate. Studies have proposed various approximation schemes that can be categorized into four groups. Let $\hat{Q}(x)$ denote the approximate function. The first group comprises scenario methods that approximate the expected recourse function by the sample average of $Q(x, \omega_i)$ for several samples: $\omega_1, \omega_2, ..., \omega_N$ (Kleywegt et al., 2001; Long et al., 2012). The approximation function is successively updated as follows:

$$\hat{Q}(x) = \frac{\sum_{i=1}^{N} Q(x, \omega_i)}{N}.$$

Generally, the method is very efficient, but the solution might not always converge to the optimal solution.

The second group includes stochastic gradient techniques (Ermoliev, 1988; Guigues, 2017b; Nemirovski et al., 2009; Polyak and Juditsky, 1992; Robbins and Monro, 1951). This type of method updates solutions by using stochastic subgradients as directions. The approximate function is successively updated as follows:

$$\hat{Q}(x) = (\bar{g}^k)^T x, \tag{3}$$

where $\bar{g}^k$ denotes a smoothed estimate of the gradient of the expected recourse function at $x$ for iteration $k$. Despite being inefficient, this method has been proven to be convergent by projection (Rockafellar and Wets, 1988) or recursive linearization (Ruszczyński, 1987).

The third group mainly consists of *primal* and *dual* decomposition methods. The use of *primal* and *dual* decomposition methods dates back to Benders decomposition (Benders, 1962). Van Slyke and Wets (1969) first described the application of Benders decomposition to two-stage stochastic programs with the L-shaped algorithm, followed by Higle and Sen (1991) with their stochastic decomposition method. Furthermore, one algorithm that has been widely applied is the stochastic dual dynamic programming (SDDP) by Pereira and Pinto (1991). This algorithm constructs feasible dynamic programming policies by using an outer approximation of a (convex) recourse function computed using Benders cuts. The SDDP algorithm has led to a number of related methods that are based on the same essential idea but seek to improve the method by exploiting the structure of particular applications. These methods include risk-averse variants (Guigues and Römisch, 2012; Philpott and de Matos, 2012; Shapiro, 2011; Shapiro et al., 2013), adaptation of SDDP for problems with interstage dependent processes (Guigues, 2014; Infanger and Morton, 1996; Lohmann et al., 2016), use of inexact cuts (Guigues, 2017a; Guigues, 2018; Zakeri et al., 2000), embedding of SDDP in the scenario tree framework (Rebennack, 2016), combination of SDDP and Lagrangian relaxation (Steeger et al., 2018), and approximate dual dynamic programming (Löhndorf et al., 2013). The convergence of SDDP and related methods was proven by Philpott and Guan (2008) for linear programs, by Girardeau et al. (2015) for risk-neutral nonlinear problems, and by Guigues (2016) for risk-averse nonlinear problems.

The fourth group consists of separable approximation methods (Cheung and Powell, 2000; Powell et al., 2004; Nascimento and Powell, 2013). These methods replace the expected recourse function in Equation (1) with the following separable approximation functions:

$$\hat{Q}(x) = \sum_{i=1}^{I} \hat{Q}_i(x). \tag{4}$$

If the separable functions $\hat{Q}_i(x)$ are linear or piecewise linear, then the expected recourse function can be replaced in Equation (1) with $\hat{Q}(x)$, and the problem can be solved as a pure network flow problem for network recourse problems, which is a well-known polynomial solvable problem. For example, Godfrey and Powell (2002) proposed the adaptive piecewise concave approximation (CAVE) algorithm, which demonstrated exceptionally good experimental performance, but they did not provide any provable convergent results. To achieve convergence to optimal solutions, Cheung and Powell (2000) proposed an approximation algorithm (SHAPE) by using a sequence of strongly convex approximation functions. Strong convexity requires a nonlinear term in the approximation functions that destroys the pure network structure and demands additional computational effort. In this study, we attempt to combine our ability to construct accurate and efficient approximations with the convergence property.

## *1.3. Contributions*

We aim to develop a *convergent* method that can *efficiently* approximate the expected recourse function for two-stage stochastic programs. The main objectives of this study are as follows:

1. We propose a new convergent hybrid learning algorithm to approximate the expected recourse function for two-stage stochastic programs.
2. We theoretically prove the convergence of the algorithm for general two-stage stochastic programs. The results indicate that when initial piecewise linear approximation functions are properly constructed, the algorithm can drop the projection step without sacrificing the convergence for two-stage stochastic programs with network recourse. Therefore, a pure piecewise linear approximation method is indeed convergent. This interesting finding answers the following question that has puzzled researchers for more than a decade: Why does piecewise linear approximation work well for two-stage stochastic programs? Our analytical results provide the first theoretical support for the use of the piecewise linear approximation method in two-stage stochastic programs.
3. We conduct a performance analysis. The computation results reveal the efficiency of the proposed algorithms, which are distribution-free. In this study, the initial function we use for piecewise linear approximation has an equal difference between every two consecutive breakpoints, and we refer to this difference as its granularity. Our computational results also show that the granularity of the initial approximation function ($\delta$) affects the convergence rate. A small granularity of the initial function leads to a high convergence rate. Finally, the proposed algorithm is competitive for high-dimensional problems.

The rest of this paper is organized as follows. Section 2 shows the description and convergence analysis of the algorithm for general two-stage stochastic programs. Section 3 presents the algorithm (without projection steps) for two-stage stochastic programs with network recourse. Section 4 reports numerical experiments of performance analysis. Finally, we present the conclusions and outline directions for future research in Section 5.

## 2. Description and convergence analysis of ProSHLA for general two-stage stochastic programs

We introduce ProSHLA and then conduct a convergence analysis of ProSHLA for general two-stage stochastic programs in this section.

### *2.1. Description of ProSHLA*

To present ProSHLA mathematically, at each iteration $k$, we let

| | | |
|---|---|---|
| $a_k$ | = | (possibly random) positive step size; |
| $\bar{Q}(x)$ | = | expected recourse function, that is, $E_\omega[Q(x,\omega)]$; |
| $\hat{Q}^k(x)$ | = | convex differentiable approximation of $\bar{Q}(x)$; |
| $g^k$ | = | stochastic subgradient of $\bar{Q}(x)$ at $x^k$, that is, $g^k \in \partial Q(x^k, \omega^{k+1})$; |
| $\bar{g}^k$ | = | smoothed estimate of the gradient of $\bar{Q}(x)$ at iteration $k$; |
| $\hat{q}^k(x)$ | = | subgradient of $\hat{Q}^k(x)$ at $x$, that is, $\hat{q}^k(x) \in \partial \hat{Q}^k(x)$; and |
| $\mathcal{H}_k = \{\omega_0, \omega_1, ..., \omega_k\}$ | = | the history up to (and including) iteration $k$. |

For a general non-smooth convex function $\hat{Q}(x)$, we define its sub-differential as follows:

$$\partial \hat{Q}(x) = \{\hat{q}(x) \in \mathbb{R}^n : \hat{Q}(y) - \hat{Q}(x) \geq \hat{q}(x)^T (y - x)\}.$$

In this study, we combine Equations (1), (3), and (4) to form an approximation at iteration $k$ as follows:

$$\min_{x \in X} c_0^T x + \hat{Q}^0(x) + (\bar{g}^k)^T x. \tag{5}$$

We approximate the expected recourse function at iteration $k$ through a convex, differentiable approximation $\hat{Q}^0(x)$ with a linear correction term $(\bar{g}^k)^T x$. At each iteration, we use the linear correction term $(\bar{g}^k)^T x$ to improve the initial approximation $\hat{Q}^0(x)$. Note that we use a convex initial approximation function $\hat{Q}^0(x)$ here, whereas SHAPE uses strongly convex approximation functions. Strong convexity requires a nonlinear term in the approximation function that may destroy the pure network flow problem structure and demands additional computational effort.

However, we do not calculate $\bar{g}^k$ in the usual manner to obtain stochastic subgradients in this study. Instead, our model uses the following form:

$$\min_{x \in X} c_0^T x + \hat{Q}^k(x) + a_k(g^k - \hat{q}^k(x^k))^T x, \tag{6}$$

where $\hat{Q}^k(x)$ is updated as follows:

$$\hat{Q}^{k+1}(x) = \hat{Q}^k(x) + a_k(g^k - \hat{q}^k(x^k))^T x. \tag{7}$$

The advantage of updating $\hat{Q}^{k+1}(x)$ is that the stochastic subgradients used in the previous iterations, namely, $(\hat{q}^k(x^k), \hat{q}^{k-1}(x^{k-1}), ..., \hat{q}^0(x^0))$, can be retained. Thus, the objective function in iteration $k$ involves a weighted average of stochastic subgradients in previous $(k-1)$ iterations. As shown later in Lemma 2, $\bar{g}^k$ in Equation (5) is a linear combination of $g^1, g^2, ..., g^{k-1}$.

Let $P_X : R^n \to X$ be the orthogonal projection onto $X$ (Rockafellar and Wets, 1988). Subsequently, we generate a sequence of solutions $\{x^k\}$ through the following steps:

---

**ProSHLA**

**Step 1.** Set iteration counter $k = 0$ and pass counter $m = 0$. Construct an initial *convex* function $\hat{Q}^0(x)$. Maintain a sequence of points: $\{x^k\}$. Let $\overline{m}$ be the number of the first iteration of the $m$ pass.

**Step 2.** Obtain $x^0 = \arg\min_{x \in X} \hat{Q}^0(x)$.

**Step 3.** Obtain $\hat{q}^k(x^k)$ and $g^k$. Let $\overline{m} = k$, $x^{\overline{m}} = x^k$, and $\hat{q}^{\overline{m}}(x^{\overline{m}}) = \hat{q}^k(x^k)$. Update $\hat{Q}^{k+1}(x)$ by

$$\hat{Q}^{k+1}(x) = \hat{Q}^k(x) + a_k(g^k - \hat{q}^k(x^k))^T x.$$

**Step 4.** Obtain $x^{k+1}$ by

$$x^{k+1} = \arg\min_{x \in X} \hat{Q}^{k+1}(x). \tag{8}$$

**Step 5.** If $|\hat{q}^{\overline{m}}(x^{k+1}) - \hat{q}^{\overline{m}}(x^{\overline{m}})| > 0$ or $x^{k+1} = x^{\overline{m}}$, then update the pass counter $m = m + 1$ and go to step 7; otherwise, go to Step 6.

**Step 6.** Obtain $x^{k+1}$ by

$$x^{k+1} = P_X(x^k - a_k g^k). \tag{9}$$

Thereafter, go to Step 5.

**Step 7.** Check for convergence (e.g., an improvement in $\hat{Q}^k(x)$ in the last $K$ iterations). If the check fails, then set $k = k + 1$, and go to Step 3; otherwise, terminate.

---

Figure 1: **Description of ProSHLA.**

Specifically, ProSHLA maintains two-level loops. The first-level loop has a series of *passes*. The second-level loop has a series of *projection steps*, which consist of Steps 5 and 6. In the first pass, given an initial bounded and piecewise linear convex approximation function $\hat{Q}^0(x)$, ProSHLA begins by solving problem (1) and obtains the initial solution $x^0$. Subsequently, we draw a realization of the random quantity $\omega \in \Omega$, solve the resulting deterministic problem and obtain a stochastic subgradient of $\bar{Q}(x)$. We compare the stochastic subgradient with the slope of $\hat{Q}^0(x)$ at $x = x^0$. The difference of the slopes is used as a linear term to update $\hat{Q}^0(x)$. The updated approximation is used to obtain a new solution $x^{k+1}$. When the newly obtained solution $x^{k+1}$ exhibits exactly identical subgradient vectors $\hat{q}^{\overline{m}}(x^{k+1})$ as solution $x^{\overline{m}}$, the piecewise linear approximation method may jump into a local optimum. Then, from iteration $k+1$, ProSHLA enters the second-level loop and implements projection steps, in which a stochastic subgradient method is applied to jump out from the local optimum. When a new solution $x^{k+1}$ is obtained such that $\hat{q}^{\overline{m}}(x^{k+1})$ is different from $\hat{q}^{\overline{m}}(x^{\overline{m}})$, ProSHLA jumps out of the projection step and proceeds to the next pass, and the entire process is repeated. Finally, we terminate the algorithm when the total absolute change in $\hat{Q}^l(x)$ over a certain number of iterations is low (e.g., $\sum_{l=k-M+1,k>M}^{k} \parallel \hat{Q}^l(x) - \hat{Q}^{l-1}(x) \parallel < \delta$). Note that $k$ represents the iteration number, and it updates when ProSHLA starts a new iteration to obtain a new solution $x^{k+1}$ (in Steps 4 and 6); $m$ represents the number of ProSHLA crossing the projection steps, and it updates when ProSHLA satisfies the condition in Step 5; $\overline{m}$ represents the number of the first iteration of the $m$ pass, and it updates when ProSHLA enters a new pass in Step 3.

The most significant difference between SHAPE and ProSHLA is that SHAPE uses strongly convex approximation functions, whereas ProSHLA uses convex approximation functions. Strong convexity requires a nonlinear term in the approximation function that may destroy the pure network flow problem structure and demands additional computational effort. To overcome the limitation of the SHAPE algorithm, the ProSHLA algorithm introduces the projection step to help construct approximation functions, which are not strictly convex. Without this step, the ProSHLA algorithm might become stuck in the corner solution for stochastic linear programs. Thus, ProSHLA is designed to work with the piecewise convex problem that arises in stochastic linear programs.

## *2.2. Convergence analysis of ProSHLA*

In this subsection, we state the convergence theorem of ProSHLA. Subsequently, we list several properties of the approximation that are used to prove the convergence of ProSHLA. Finally, we present the proof for our theorem.

Without loss of generality, we make the following assumptions.

(A.1) $X \subset \mathbb{R}^n$ is convex and compact.

(A.2) $E_\omega Q(x, \omega)$ is convex, finite and continuous on $X$.

(A.3) $g^k$ is bounded such that $||g^k|| \leq C_1$ for each $\omega \in \Omega$; $\hat{q}^k$ is bounded such that $||\hat{q}^k|| \leq C_2$ for each $\omega \in \Omega$.

(A.4) Piecewise linear functions $\hat{Q}^k(x)$ are convex, implying that $\hat{Q}^k(x) - \hat{Q}^k(y) \leq \hat{q}^k(x)^T(x-y)$.

(A.5) The stepsizes $a_k$ are $\mathcal{H}_k$-measurable and satisfy

$$0 < a_k < 1, \quad \sum_{k=0}^{\infty} E\{a_k^2\} \leq \infty.$$

We note that assumption (A.1) holds if $Q(x, \omega)$ denotes a convex hull of a finite number of points. Generally, given that $Q(x, \omega)$ is real-valued and convex over the entire space $\mathbb{R}^n$, the sub-differential $\partial \hat{Q}^k(x)$ is nonempty and compact for all $x$. Specifically, assumption (A.3) is satisfied if each $Q(x, \omega)$ is polyhedral (i.e., it is the point-wise maximum of a finite number of affine functions).

In addition to the assumptions from (A.1) to (A.5), the following assumption is required to characterize the piecewise linear convex approximation functions.

(A.6) There exists a constant $\delta$ and a positive $b > 0$, such that for any two points $x, y \in X$, if $|x - y| > \delta$, then $|\hat{q}(x) - \hat{q}(y)| \geq b|x - y|$. If there exists $\hat{q}(x)$ and $\hat{q}(y)$ such that $\hat{q}(x) - \hat{q}(y) = 0$, then $|x - y| \leq \delta$. If $\delta \to \infty$, then the function becomes purely linear; if $\delta \to 0$, then the function corresponds to a strongly convex function.

Given assumptions (A.1) to (A.6), we obtain the following preliminary result of ProSHLA.

**Theorem 1.** *If assumptions (A.1)-(A.6) are satisfied, then the sequence of $\{x_k\}$ generated by ProSHLA converges almost surely to the optimal solution $x^* \in X^*$ of problem (1).*

*2.2.1. Properties of approximations*

We state the Martingale convergence theorem that is used to prove the convergence of some stochastic subgradient methods (Taylor, 1990).

**MARTINGALE CONVERGENCE THEOREM**. *A sequence of random variables $\{W^k\}$, which are $\mathcal{H}_k$-measurable, is said to be a supermartingale if the sequence of conditional expectations $E\{W^{k+1}|\mathcal{H}_k\}$ exists and satisfies $E\{W^{k+1}|\mathcal{H}_k\} \le W^k$.*

**Theorem 2.** (From Neveu, 1975, p.26) *Let $W^k$ be a positive supermartingale. Then, $W^k$ converges to a finite random variable a.s.*

The above definition indicates that $W^k$ is essentially the stochastic analogue of a decreasing sequence.

Notably, our algorithm adjusts the approximations by a linear term at each iteration. Thus, in conjunction with assumption (A.6), the following property is obtained:

**Proposition 1.** *For any two points $x_1$ and $x_2$, if $\hat{q}^0(x_1) = \hat{q}^0(x_2)$, then $\hat{q}^k(x_1) = \hat{q}^k(x_2)$ for any $k > 1$.*

The proposition holds because the approximation functions are updated by linear terms.

**Proposition 2.** *For any two iterations in the $m^{th}$ pass, say $x^j$ and $x^{j'}$, $\hat{q}^k(x^j) = \hat{q}^k(x^{j'}) = \hat{q}^{\overline{m}}(x^j)$ for any $\overline{m} \le k < \overline{m+1}$.*

The proposition holds because the approximation functions are not updated when the algorithm performs the projection steps.

On the basis of the convexity property, the optimal solution for problem (8) at iteration $\overline{m}$ is characterized by the following variational inequality:

$$(\hat{q}^{\overline{m}}(x^{\overline{m}}))^T(x - x^{\overline{m}}) \ge 0, \quad \forall x \in X. \tag{10}$$

To obtain Theorem 1, we require the following three lemmas. The first lemma indicates that the difference between the solutions of two consecutive passes is bounded by the stepsize and magnitude of the stochastic gradient. The second lemma shows that the approximation $\hat{Q}^k(x)$ is finite, and the third lemma indicates that $T^k$ (which is explained later) is bounded.

**Lemma 1.** *For any two iterations $i \in [\overline{m}, \overline{m+1})$ and $j \in [\overline{m+1}, \overline{m+2})$, solutions $x^i$ and $x^j$ produced by ProSHLA satisfy the following:*

$$a_{\overline{m}} g^{\overline{m}}(x^i - x^j) \leq (a_{\overline{m}} C_1)^2/(b). \tag{11}$$

*Proof.* Consider a special case, where $i$ and $j$ correspond to two consecutive iterations. Let $i = \overline{m+1} - 1$ and $j = \overline{m+1}$. On the basis of (10), we know that,

$$(\hat{q}^{\overline{m+1}}(x^{\overline{m+1}}))^T(x - x^{\overline{m+1}}) \geq 0, \quad \forall x \in X. \tag{12}$$

On the basis of the updating rule of the approximation function, we obtain

$$(\hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}}) + a_{\overline{m+1}-1}(g^{\overline{m+1}-1} - \hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}-1})))^T(x - x^{\overline{m+1}}) \geq 0, \quad \forall x \in X. \tag{13}$$

We substitute $x$ with $x^{\overline{m+1}-1}$ in (13) and obtain

$$a_{\overline{m+1}-1}(g^{\overline{m+1}-1} - \hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}-1}))^T(x^{\overline{m+1}-1} - x^{\overline{m+1}}) \geq \hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}})^T(x^{\overline{m+1}} - x^{\overline{m+1}-1}). \tag{14}$$

We rearrange the terms to obtain the following expression:

$$\begin{aligned} a_{\overline{m+1}-1}(g^{\overline{m+1}-1})^T(x^{\overline{m+1}-1} - x^{\overline{m+1}}) &\geq \hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}})^T(x^{\overline{m+1}} - x^{\overline{m+1}-1}) \\ &- a_{\overline{m+1}-1} \cdot (\hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}-1})^T(x^{\overline{m+1}} - x^{\overline{m+1}-1})) \\ &= (\hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}}) - \hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}-1}))^T(x^{\overline{m+1}} - x^{\overline{m+1}-1}) \\ &+ (1 - a_{\overline{m+1}-1})\hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}-1})^T(x^{\overline{m+1}} - x^{\overline{m+1}-1}). \end{aligned} \tag{15}$$

If iterations $\overline{m+1}-1$ and $\overline{m+1}$ are not in the same pass, and this implies that $\hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}-1}) \neq \hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}})$. Based on assumption (A.6), we known that $|\hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}-1}) - \hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}})| \geq b|x^{\overline{m+1}-1} - x^{\overline{m+1}}|$.

On the basis of Equation (10), Equation (13), and $0 < a_{\overline{m+1}-1} < 1$, we obtain

$$\begin{aligned} a_{\overline{m+1}-1}(g^{\overline{m+1}-1})^T(x^{\overline{m+1}-1} - x^{\overline{m+1}}) &\geq b||x^{\overline{m+1}-1} - x^{\overline{m+1}}||^2 \\ &+ (1 - a_{\overline{m+1}-1})\hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}-1})^T(x^{\overline{m+1}} - x^{\overline{m+1}-1}) \\ &\geq b||x^{\overline{m+1}-1} - x^{\overline{m+1}}||^2. \end{aligned}$$

We apply Schwartz's inequality to obtain the following expression:

$$\begin{aligned} a_{\overline{m+1}-1}||g^{\overline{m+1}-1}|| \cdot ||x^{\overline{m+1}-1} - x^{\overline{m+1}}|| &\geq a_{\overline{m+1}-1}(g^{\overline{m+1}-1})^T(x^{\overline{m+1}-1} - x^{\overline{m+1}}) \\ &\geq b||x^{\overline{m+1}-1} - x^{\overline{m+1}}||^2. \end{aligned}$$

Thus, $||x^{\overline{m+1}-1} - x^{\overline{m+1}}|| \leq a_{\overline{m+1}-1}C_1/b$. The following expression is applicable:

$$a_{\overline{m+1}-1}(g^{\overline{m+1}-1})^T(x^{\overline{m+1}-1} - x^{\overline{m+1}}) \leq (a_{\overline{m+1}-1}C_1)^2/b. \tag{16}$$

For any $i \in [\overline{m}, \overline{m+1} - 1]$, $g^i = g^{\overline{m}} = g^{\overline{m+1}-1}$ and $\hat{q}^i(x) = \hat{q}^{\overline{m}}(x) = \hat{q}^{\overline{m+1}-1}(x)$. Therefore,

$$a_{\overline{m}}(g^{\overline{m}})^T(x^i - x^{\overline{m+1}}) \leq (a_{\overline{m}}C_1)^2/b. \tag{17}$$

For any $i \in [\overline{m}, \overline{m+1} - 1]$ and $j \in [\overline{m+1}, \overline{m+1} - 1]$, $\hat{q}^j(x) = \hat{q}^{\overline{m+1}}(x) = \hat{q}^{\overline{m+1}-1}(x)$ for any $x \in X$. Therefore,

$$a_{\overline{m}}(g^{\overline{m}})^T(x^i - x^j) \leq (a_{\overline{m}}C_1)^2/b. \tag{18}$$

□

**Lemma 2.** *The approximation function $\hat{Q}^k(x)$ in iteration $k$ can be written as $\hat{Q}^k(x) = \hat{Q}^0(x) + (\bar{g}^k)^T x$, where $\bar{g}^k$ is a finite vector.*

*Proof.* According to Equation (5), the approximation in iteration $k$ is the initial approximation $\hat{Q}^0(x)$ plus a linear term as follows:

$$\hat{Q}^k(x) = \hat{Q}^0(x) + (\bar{g}^k)^T x. \tag{19}$$

Here, $(\bar{g}^k)^T x$ shows the cumulative change of $\hat{Q}^0$ up to iteration $k$. We will demonstrate below that $\bar{g}^k$ is a finite vector for any $k$.

Taking the first derivative of $\hat{Q}^k(x)$ in Equation (19), we have

$$\hat{q}^k(x) = \hat{q}^0(x) + \bar{g}^k. \tag{20}$$

On the basis of Equations (7) and (20), we can obtain $\hat{Q}^{k+1}(x)$ as follows:

$$\begin{aligned}\hat{Q}^{k+1}(x) &= \hat{Q}^k(x) + a_k(g^k - \hat{q}^k(x))^T x \\ &= \hat{Q}^0(x) + (\bar{g}^k)^T x + a_k(g^k - \hat{q}^k(x))^T x \\ &= \hat{Q}^0(x) + (\bar{g}^k)^T x + a_k(g^k - \hat{q}^0(x) + \bar{g}^k)^T x.\end{aligned}$$

Thus, we can obtain the relationship between $\bar{g}^{k+1}$ and $\bar{g}^k$ as follows:

$$\bar{g}^{k+1} = a_k(g^k - \hat{q}^0(x_k)) + (1 - a_k)\bar{g}^k. \tag{21}$$

From the Equation (21), we conclude that $\bar{g}^{k+1}$ is a linear combination of $g^1, g^2, ..., g^k$. Since $g^k$ and $\hat{q}^0(x_k)$ are finite, there exists a finite and positive vector $\hat{d}$ such that

$$\hat{d} \geq \max_k |g^k - \hat{q}^0(x_k)|, \tag{22}$$

where the inequality is applied componentwise.

According to Lemma 2 in Cheung and Powell (2000), we can conclude that $\bar{g}^{k+1} \leq \hat{d}$. □

Let $T^k = \hat{Q}^k(x^*) - \hat{Q}^k(x^k)$, where $x^*$ denotes the optimal solution. The following lemma characterizes the difference between $T^{k+1}$ and $T^k$.

**Lemma 3.** *For any two iterations $i \in [\overline{m-1}, \overline{m} - 1]$ and $j \in [\overline{m}, \overline{m+1} - 1]$, $T^i$ and $T^j$ satisfy*

$$T^j - T^i \leq a_{\overline{m}}(g^{\overline{m}})^T(x^i - x^j) + a_{\overline{m}}(g^{\overline{m}})(x^* - x^i). \tag{23}$$

*Proof.* We consider a special case. Let $i = \overline{m}$ and $j = \overline{m+1}$. By re-writing $x^* - x^{\overline{m+1}}$ as $x^* - x^{\overline{m}} + x^{\overline{m}} - x^{\overline{m+1}}$, we obtain

$$\begin{aligned}\hat{Q}^{\overline{m+1}}(x) &= \hat{Q}^{\overline{m+1}-1}(x) + a_{\overline{m+1}-1}(g^{\overline{m+1}-1} - \hat{q}^{\overline{m+1}-1}(x^{\overline{m+1}-1}))^T x \\ &= \hat{Q}^{\overline{m}}(x) + a_{\overline{m}}(g^{\overline{m}} - \hat{q}^{\overline{m}}(x^{\overline{m}}))^T x.\end{aligned}$$

Subsequently,

$$\begin{aligned}
T^{\overline{m+1}} - T^{\overline{m}} &= \hat{Q}^{\overline{m}}(x^*) + a_{\overline{m}}(g^{\overline{m}} - \hat{q}^{\overline{m}}(x^{\overline{m}}))^T x^* - \\
&\quad (\hat{Q}^{\overline{m}}(x^{\overline{m+1}}) + a_{\overline{m}}(g^{\overline{m}} - \hat{q}^{\overline{m}}(x^{\overline{m}})^T x^{\overline{m+1}})) - (\hat{Q}^{\overline{m}}(x^*) - \hat{Q}^{\overline{m}}(x^{\overline{m}})) \\
&= a_{\overline{m}}(g^{\overline{m}} - \hat{q}^{\overline{m}})^T(x^* - x^{\overline{m}} + x^{\overline{m}} - x^{\overline{m+1}}) + \hat{Q}^{\overline{m}}(x^{\overline{m}}) - \hat{Q}^{\overline{m}}(x^{\overline{m+1}}) \\
&= \underbrace{(\hat{Q}^{\overline{m}}(x^{\overline{m}}) - \hat{Q}^{\overline{m}}(x^{\overline{m+1}}) - a_{\overline{m}}(\hat{q}^{\overline{m}})^T(x^{\overline{m}} - x^{\overline{m+1}}))}_{I} - \underbrace{a_{\overline{m}}(\hat{q}^{\overline{m}})^T(x^* - x^{\overline{m}})}_{II} \\
&+ \underbrace{a_{\overline{m}}(g^{\overline{m}})^T(x^{\overline{m}} - x^{\overline{m+1}})}_{III} + \underbrace{a_{\overline{m}}(g^{\overline{m}})^T(x^* - x^{\overline{m}})}_{IV}.
\end{aligned}$$

We consider each part individually. Given that $\hat{q}^{\overline{m}} \in \partial\hat{Q}^{\overline{m}}(x^{\overline{m}})$, by convexity of $\hat{Q}^{\overline{m}}(x)$, we obtain

$$\hat{Q}^{\overline{m}}(x^{\overline{m}}) - \hat{Q}^{\overline{m}}(x^{\overline{m+1}}) \le (\hat{q}^{\overline{m}})^T(x^{\overline{m}} - x^{\overline{m+1}}). \tag{24}$$

Hence, the following expression is applicable.

$$\begin{aligned}
\hat{Q}^{\overline{m}}(x^{\overline{m}}) - \hat{Q}^{\overline{m}}(x^{\overline{m+1}}) &\le (\hat{q}^{\overline{m}})^T(x^{\overline{m}} - x^{\overline{m+1}}) && (25)\\
&= (1 - a_{\overline{m}})(\hat{q}^{\overline{m}})^T(x^{\overline{m}} - x^{\overline{m+1}}) + a_{\overline{m}}(\hat{q}^{\overline{m}})^T(x^{\overline{m}} - x^{\overline{m+1}}). && (26)
\end{aligned}$$

Given Equation (10) and $0 < a_{\overline{m}} < 1$, we know that $(I) \le 0$. Additionally, from Equation (10) and $0 < a_{\overline{m}} < 1$, we know that $(II) \ge 0$. Thus, $T^{\overline{m+1}} - T^{\overline{m}} \le a_{\overline{m}}(g^{\overline{m}})^T(x^{\overline{m}} - x^{\overline{m+1}}) + a_{\overline{m}}(g^{\overline{m}})(x^* - x^{\overline{m}})$.

For any $i \in [\overline{m}, \overline{m+1} - 1]$, $g^i = g^{\overline{m}} = g^{\overline{m+1}-1}$ and $\hat{q}^i(x) = \hat{q}^{\overline{m}}(x) = \hat{q}^{\overline{m+1}-1}(x)$ for any $x \in X$. Therefore,

$$T^{\overline{m+1}} - T^i \le a_{\overline{m}}(g^{\overline{m}})^T(x^i - x^{\overline{m+1}}) + a_{\overline{m}}(g^{\overline{m}})(x^* - x^i). \tag{27}$$

For any $i \in [\overline{m}, \overline{m+1} - 1]$ and $j \in [\overline{m+1}, \overline{m+2} - 1]$, $\hat{Q}^j(x) = \hat{Q}^{\overline{m+1}}(x) = \hat{Q}^{\overline{m+2}-1}(x)$ for any $x \in X$. Therefore,

$$T^j - T^i \le a_{\overline{m}}(g^{\overline{m}})^T(x^i - x^j) + a_{\overline{m}}(g^{\overline{m}})(x^* - x^i). \tag{28}$$

□

### *2.2.2. Proof of Theorem 1*

With regard to our main result, we consider two cases. In the first case, the algorithm does not terminate in a given pass. Therefore, any pass before the algorithm terminates exhibits finite iterations, that is, $\overline{m+1} - \overline{m} < M$ for any $m$, where $M$ denotes a large number. In the second case, the algorithm may halt in a given pass. We prove Theorem 1 in each case.

*Case 1: The algorithm does not terminate in a given pass.*

In this case, we consider a subsequence of $\{x^k\}$, $\{x^{\overline{m}}\}$. We prove that the subsequence $\{x^{\overline{m}}\}$ converges to the true optimal $x^*$.

By the definition of $g^k \in \partial Q(x^k, w^{k+1})$,

$$(g^k)^T(x^* - x^k) \quad \leq \quad Q(x^*, w^{k+1}) - Q(x^k, w^{k+1}), \tag{29}$$

where $Q(x, w^{k+1})$ denotes the recourse function given the outcome $w^{k+1}$.

Lemma 1 implies the following:

$$a_{\overline{m}} g^{\overline{m}}(x^* - x^{\overline{m}}) \leq (a_{\overline{m}} C_1)^2/(b). \tag{30}$$

Based on Lemma 3, the difference $T^{\overline{m+1}} - T^{\overline{m}}$ is as follows:

$$\begin{aligned} T^{\overline{m+1}} - T^{\overline{m}} \quad &= \quad a_{\overline{m}}(g^{\overline{m}})^T(x^{\overline{m}} - x^{\overline{m+1}}) + a_{\overline{m}}(g^{\overline{m}})^T(x^* - x^{\overline{m}}) && (31) \\ &\leq -a_{\overline{m}}(Q(x^{\overline{m}}, w^{\overline{m+1}}) - Q(x^*, w^{\overline{m+1}})) + a_{\overline{m}}(g^{\overline{m}})^T(x^* - x^{\overline{m}}) && (32) \\ &\leq -a_{\overline{m}}(Q(x^{\overline{m}}, w^{\overline{m+1}}) - Q(x^*, w^{\overline{m+1}})) + (a_{\overline{m}} C_1)^2/(b). && (33) \end{aligned}$$

We take conditional expectation with respect to $\mathcal{H}_k$ on both sides and obtain

$$E\{T^{\overline{m+1}} | \mathcal{H}_{\overline{m}}\} \leq T^{\overline{m}} - a_{\overline{m}}(\bar{Q}(x^{\overline{m}}) - \bar{Q}(x^*)) + (C_1 a_{\overline{m}})^2/(b), \tag{34}$$

where $T^{\overline{m}}$, $a_{\overline{m}}$ and $x^{\overline{m}}$ on the right-hand side are deterministic given the conditioning on $\mathcal{H}_k$. The conditioning on $\mathcal{H}_k$ does not provide any information on $w^{k+1}$. Thus, we replace $Q(x, w^{\overline{m+1}})$ (for

$x = x^k$ and $x = x^*$) with its expectation $\bar{Q}(x)$. Given that $a_{\overline{m}}(\bar{Q}(x^{\overline{m}}) - \bar{Q}(x^*)) \geq 0$, the sequence

$$W^{\overline{m}} = T^{\overline{m}} + (C_1^2/b) \sum_{i=\overline{m}}^{\infty} a_i^2 \tag{35}$$

is a positive supermartingale. Theorem 2 implies the almost sure convergence of $W^{\overline{m}}$. Thus,

$$T^{\overline{m}} \to T^* \quad a.s. \tag{36}$$

We perform the summation of (33) from 0 to $\overline{M}$ and obtain the following expression:

$$T^{\overline{M+1}} - T^0 \leq -\sum_{\overline{m}=0}^{\overline{M}} a_{\overline{m}}(Q(x^{\overline{m}}, w^{\overline{m+1}}) - Q(x^*, w^{\overline{m+1}})) + \sum_{\overline{m}=0}^{\overline{M}} (a_{\overline{m}} C_1)^2/b. \tag{37}$$

We take the expectations of both sides. For the first term on the right-hand side, we take the conditional expectation with respect to $\mathcal{H}_{\overline{m}}$ and then over all $\mathcal{H}_{\overline{m}}$.

$$\begin{aligned} E\{T^{\overline{M+1}} - T^0\} &\leq -\sum_{\overline{m}=0}^{\overline{M}} E\{E\{a_{\overline{m}}(Q(x^{\overline{m}}, \omega^{\overline{m+1}}) - Q(x^*, \omega^{\overline{m+1}}))|\mathcal{H}_{\overline{m}}\}\} + E\{\sum_{\overline{m}=0}^{\overline{M}} (C_1 a_{\overline{m}})^2/(b)\} \\ &\leq -\sum_{\overline{m}=0}^{\overline{M}} E\{a_{\overline{m}}(\bar{Q}(x^{\overline{m}}) - \bar{Q}(x^*))|\mathcal{H}_{\overline{m}}\} + C_1^2/b \sum_{\overline{m}=0}^{\overline{M}} E\{a_{\overline{m}}^2\}. \end{aligned} \tag{38}$$

We take the limit as $\overline{M} \to \infty$ and use the finiteness of $T^{\overline{m}}$ and $\sum_{\overline{m}=0}^{\infty} E\{a_{\overline{m}}^2\}$ to obtain

$$\sum_{\overline{m}=0}^{\overline{M}} E\{a_{\overline{m}}(\bar{Q}(x^{\overline{m}}) - \bar{Q}(x^*))|\mathcal{H}_{\overline{m}}\} < \infty. \tag{39}$$

Given that $Q(x^{\overline{m}}, \omega^{m+1}) - Q(x^*, \omega^{m+1}) \geq 0$ and $\sum_{\overline{m}=0}^{\infty} a_{\overline{m}} = \infty$ (a.s.), there exists a subsequence $\{\overline{m}\}$ such that

$$\bar{Q}(x^{\overline{m}}) \to \bar{Q}(x^*) \quad a.s.$$

By continuity of $\overline{Q}$, the sequence converges. Thus,

$$x^{\overline{m}} \to x^* \quad a.s. \tag{40}$$

Then, we construct another subsequence $\{x^{\overline{m}-1}\}$. Based on (33),

$$T^{\overline{m+2}-1} - T^{\overline{m+1}-1} \leq -a_{\overline{m}}(Q(x^{\overline{m+1}-1}, w^{\overline{m+2}-1}) - Q(x^*, w^{\overline{m+2}-1})) + (C_1 a_{\overline{m}})^2/(b). \tag{41}$$

Similarly, we can prove the following:

$$x^{\overline{m}-1} \to x^* \quad a.s.$$

By similar logic, we can show that a very general subsequence $\{x^i\}$, $i \in [\overline{m}, \overline{m+1}-1]$ converges to $x^*$ almost surely. We call the class of subsequence $X_s$.

All passes include finite iterations. Therefore, given any subsequence of $x^k$, we can extract a subsequence that belongs to $X_s$. Subsequently, we reach the following conclusion:

$$x^k \to x^* \quad a.s.$$

*Case 2: The algorithm terminates in a given pass.*

In this case, the algorithm terminates at a purely projected stochastic gradient procedure that produces a convergent sequence.

Therefore, we reach the conclusion. ■

The results indicate that ProSHLA is convergent. As indicated in the analysis above, the approximation function $\hat{Q}(x)$ is required to be piecewise linear convex. We are interested in a special case in which separable functions are used to approximate the expected recourse function for two-stage stochastic programs with network recourse. According to Equation (4), if the separable functions are linear or piecewise linear, then problem (1) can be solved as a pure network flow problem for the network recourse problem, which is a well-known polynomial solvable problem.

## 3. ProSHLA using separable piecewise linear functions for two-stage stochastic programs with network recourse

If $\hat{Q}(x)$ is separable for two-stage stochastic programs with network recourse, then we can further simplify ProSHLA by dropping the projection steps. The *stochastic hybrid learning algorithm* (SHLA) is the simplified version of ProSHLA and is described as follows.

Generally, SHLA is not convergent. However, when it is applied to two-stage stochastic programs with network recourse, it enjoys several advantages as follows: (1) at each iteration,

**SHLA**

**Step 1.** Set the iteration counter $k = 0$. Construct an initial piecewise linear *convex* function $\hat{Q}^0(x)$. Maintain a sequence of points $\{x^k\}$.

**Step 2.** Solve the problem $x^k = \arg\min_{x \in X} \hat{Q}^k(x)$ and obtain $\hat{q}^k(x^k)$.

**Step 3.** Obtain $g^k$. Update $\hat{Q}^k(x)$ by

$$\hat{Q}^{k+1}(x) = \hat{Q}^k(x) + a_k(g^k - \hat{q}^k(x^k))^T x. \tag{42}$$

**Step 4.** Check for termination (e.g., an improvement in $\hat{Q}^k(x)$ in the last $K$ iterations). If the check fails, then set $k = k + 1$ and go to Step 2; otherwise, terminate.

Figure 2: **Description of SHLA.**

problem $Q(x, \omega)$ is a simple network flow problem that can be solved by polynomial algorithms, and (2) the solution of $Q(x, \omega)$ is naturally integer.

When separable functions are used, assumption (A.6) is easily satisfied by artificially setting the expression, as follows:

$$\hat{q}_i^0(x_i) < \hat{q}_i^0(x_i + \delta).$$

Notably, our algorithmic strategy allows flexibility in choosing initial approximation functions with different values of $\delta$. Assuming that we set $\delta = 1$ for $i = 1, .., n$, we can guarantee

(A.7) $\hat{q}_i^0(x_i) < \hat{q}_i^0(x_i + 1)$.

Subsequently, we can reach the following theorem:

**Theorem 3.** *Given the additional assumption (A.7), SHLA is convergent for problems with network recourses.*

*Proof.* Based on assumption (A.7), with respect to any two different $x, y \in X$,

$$\hat{q}^k(x) \neq \hat{q}^k(y).$$

Therefore,

$$|\hat{q}^k(x) - \hat{q}^k(y)| > 0.$$

If we apply ProSHLA ($\delta = 1$) with such separable piecewise linear functions to two-stage stochastic programs with network recourse, then it never proceeds to the projection steps. In this case, ProSHLA is equivalent to SHLA. Thus, SHLA is convergent. ■

The analysis provides theoretical support for SHLA-type algorithms that are applied to operational management applications. Compared with SHAPE, SHLA does not require nonlinear functions and can thus solve each subproblem efficiently. Furthermore, SHLA does not require maintaining the convexity of the approximation functions because convexity is automatically maintained when the initial piecewise linear functions are properly constructed.

## 4. Experimental results of performance analysis

To evaluate the performance of the algorithms, two experimental designs are investigated: (1) an empty container repositioning problem that arises in the context of two-stage stochastic programs with network recourse, and (2) a high-dimensional resource allocation problem. All experiments are implemented on a PC with an Intel Core i7 CPU at 3.5 GHZ and 16 GB of RAM and ILOG CPLEX 12.6. Through the numerical experiments on the empty container repositioning problem, we demonstrate the effectiveness and efficiency of ProSHLA and SHLA, study the convergence performance of ProSHLA and SHLA, as well as examine how $\delta$ affects the convergence performance, and how robust ProSHLA and SHLA perform under different distributions of random demands. According to the setting of the empty container repositioning problem in actual practice, the number of ports in the container network can only be set to 40 in maximum. Therefore, in order to examine the performance of ProSHLA for larger sized instances, we have further conducted numerical experiments on a high-dimensional resource allocation problem for a warehouse-retailer network with 100 locations.

### *4.1. Problem generator for empty container repositioning problem*

We consider the application motivated by the empty container repositioning problem faced by a major Chinese freight forwarder. The forwarder needs to manage several empty containers in a port network that is physically located in the Pearl River Delta (PRD) region of Southern China.

In contrast to the empty container repositioning problem of a shipping liner in which the containers are repositioned in a set of ports on a fixed route (Song and Dong, 2008), in our application, the empty containers are actually repositioned in a port network that consists of several hubs (large terminals) and spokes (small ports). The demand for the empty containers is uncertain. Therefore, the problem belongs to a class of dynamic fleet management problems that is formulated as two-stage stochastic programs with network recourse. To formally describe the problem, we define the following notations:

$\mathcal{L}$ = set of ports,

$s_i$ = initial number of empty containers at Port $i$,

$S_i$ = number of empty containers at Port $i$ in Stage 2,

$r_{ij}$ = profit for moving a laden container from Port $i$ to Port $j$,

$c_{ij}$ = cost for moving an empty container from Port $i$ to Port $j$,

$d_{ij}$ = demand from Port $i$ to Port $j$ in Stage 1,

$D_{ij}$ = demand from Port $i$ to Port $j$ in Stage 2,

$x_{ij}$ = number of laden containers shipped from Port $i$ to Port $j$ in Stage 1,

$y_{ij}$ = number of empty containers shipped from Port $i$ to Port $j$ in Stage 1,

$x_{ij}(\omega)$ = number of laden containers shipped from Port $i$ to Port $j$ for sample $\omega$ in Stage 2,

$y_{ij}(\omega)$ = number of empty containers shipped from Port $i$ to Port $j$ for sample $\omega$ in Stage 2.

Subsequently, we formulate the problem as follows:

$$\min_{x,y} \sum_{i \in \mathcal{L}} \sum_{j \in \mathcal{L}} \{-r_{ij}x_{ij} + c_{ij}y_{ij}\} + E_\omega[Q(x,\omega)], \tag{43}$$

subject to

$$\begin{aligned}
\sum_{j}(x_{ij}+y_{ij}) &= s_i, \quad \forall i \in \mathcal{L} && (44)\\
\sum_{i}(x_{ij}+y_{ij}) &= S_j, \quad \forall j \in \mathcal{L} && (45)\\
0 \le x_{ij} &\le d_{ij}, \forall i, j \in \mathcal{L} && (46)\\
y_{ij} &\ge 0, \forall i, j \in \mathcal{L}, && (47)
\end{aligned}$$

where the recourse function $Q(x, \omega)$ is given as follows:

$$Q(x, \omega) = \min_{x(\omega), y(\omega)} \sum_{i \in \mathcal{L}} \sum_{j \in \mathcal{L}} \{-r_{ij} x_{ij}(\omega) + c_{ij} y_{ij}(\omega)\}, \tag{48}$$

subject to

$$\begin{aligned}
\sum_{j \in \mathcal{L}}(x_{ij}(\omega)+y_{ij}(\omega)) &= S_i, \quad \forall i \in \mathcal{L} && (49)\\
0 \le x_{ij}(\omega) &\le D_{ij}(\omega), \forall i, j \in \mathcal{L} && (50)\\
y_{ij}(\omega) &\ge 0. \forall i, j \in \mathcal{L}. && (51)
\end{aligned}$$

We create a set of problem instances to evaluate the algorithm. The problem generator creates ports in $\mathcal{L}$ in a 100 mile by 100 mile rectangle. We simply use the Euclidean distance between each pair of ports as the corresponding travel distance. The net profit for a demand is set to 500 cents per mile. The empty cost is set to 40 cents per mile. The holding cost for a demand is set to 15 cents per time instance. The demand $D_{ij}$ between locations $i$ and $j$ is set as follows:

$$D_{ij} = in_i \cdot out_j \cdot v,$$

where

$in_i$ = inbound potential for Port $i$;

$out_j$ = outbound potential for Port $j$;

$v$ = random variable.

The inbound and outbound potentials for each location capture the capability of the port to attract inbound flows or generate outbound flows. The inbound potential for Port $i$, $in_i$, is drawn uniformly between 0.2 and 1.8. The corresponding outbound potential $out_i$ is obtained by $out_i = 2 - in_i$. Therefore, these two potentials are negatively correlated. The motivation for the setting is partly because real-world application regions with large inbound flows typically exhibit small outbound flows. To capture the randomness in demand, we also include a random number $v$ with mean 30, that is, the typical daily demand between each pair of locations. To evaluate the effects of different distributions of the random demand, we consider three types of distributions: normal, exponential, and uniform. The step-sizes $a_k$ are set to $1/k$.

To construct initial piecewise linear functions, we solve a deterministic network flow problem in which random demands are replaced by their mean values. Subsequently, we obtain $\bar{S} = \{\bar{S}_1, \bar{S}_2, ..., \bar{S}_n\}$. For each $i \in \mathcal{L}$, we generate the initial approximation function $\hat{Q}_i^0(x)$ by

$$\hat{Q}_i^0(x) = c(x - \bar{S}_i)^2, x = 0, \delta, ...., k\delta, ..., K\delta,$$

where $c$ is a positive parameter and $x \in [0, K\delta]$. In the projection step, we solve a least-squares problem as follows:

$$x^{k+1} = \arg\min(x^{k+1} - (x_k + a_k g^k(x^k)))^2, x^{k+1} \in X.$$

## *4.2. Effectiveness and efficiency performance*

We use the posterior bound (PB), a myopic algorithm, the L-shaped algorithm (Van Slyke and Wets, 1969), and the inexact cut algorithm (Zakeri et al., 2000) as benchmarks to test the effectiveness and efficiency of the algorithms. We need to solve a deterministic network flow problem with all realized demands to obtain the PB. Such a posterior optimization involves no uncertainty because decisions with anticipation of future demands are allowed. Therefore, the cost of PB is low and normally unreachable, and it is used as the lower bound (Cheung and Powell, 1996). The myopic algorithm simply solves a static deterministic assignment problem at the current stage while ignoring uncertainties in the second stage. For the L-shaped and inexact cut algorithms, a group of linear programming problems with a series of cutting planes must be solved.

In the experiment, we consider eight instances, in which the number of ports is increased from 5 to 40 and the number of empty containers is increased from 400 to 3200. In each instance, we run 2000 samples and obtain the sample means of PB. We also obtain the solutions of the myopic, L-shaped, inexact cut algorithms, as well as SHLA and ProSHLA. For SHLA, we select two classes of initial functions with $\delta = 1$ and $\delta = 2$. For ProSHLA, we select $\delta = 2$.

Table 1: **Total cost for SHLA and ProSHLA.**

| | | Total cost (dollars) | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $\lvert\mathcal{L}\rvert$ | $N_R$ | PB | Myopic | L-shaped | Inexact cut | SHLA-1 | SHLA-2 | ProSHLA |
| 5 | 400 | -28551302 | -27761331 | -28551274 | -28551027 | -28464248 | -28463941 | -28464002 |
| 10 | 800 | -59397423 | -58432159 | -59396702 | -59396219 | -59338653 | -59338190 | -59338341 |
| 15 | 1200 | -98451193 | -93188576 | -98449868 | -98449027 | -98257484 | -98257244 | -98257395 |
| 20 | 1600 | -147390005 | -141062187 | -147388360 | -147386932 | -147269239 | -147269212 | -147269213 |
| 25 | 2000 | -185223883 | -180875614 | -185220423 | -185218963 | -185092289 | -185092113 | -185092143 |
| 30 | 2400 | -234005740 | -226978090 | -234004427 | -234002713 | -233401849 | -233401516 | -233401658 |
| 35 | 2800 | -266375728 | -260966356 | -266375468 | -266373497 | -266337862 | -266337649 | -266337660 |
| 40 | 3200 | -304910355 | -293882881 | -304908597 | -304907329 | -304907153 | -304906829 | -304906962 |

The experiment results on total cost are shown in Table 1. Column 1 of Table 1 shows the number of ports while Column 2 presents the number of empty containers. Column 3 contains the PB. Columns 4 to 9 contain the solutions achieved by the myopic, L-shaped, and inexact cut algorithms, as well as SHLA-1, SHLA-2, and ProSHLA. The results demonstrate that the solutions of the L-shaped algorithm can achieve optimal solutions, which are very close to the PB (within 0.002% gap from PB). The solutions of the inexact cut algorithm, SHLA, and ProSHLA are better than those of the myopic algorithm. Furthermore, SHLA ($\delta = 1$) produces better results than SHLA ($\delta = 2$) and ProSHLA ($\delta = 2$) because a small $\delta$ leads to good solutions. We will have a special discussion with the impact of $\delta$ later in the following subsection. Meanwhile, ProSHLA ($\delta = 2$) slightly outperforms SHLA ($\delta = 2$). The reason is that the projection steps in ProSHLA improve the solution. As the speed of convergence is quite important in practical problems, our focus is on the computation time for different algorithms (Table 2).

As shown in Table 2, SHLA and ProSHLA are more efficient than the L-shaped and inexact cut algorithms because SHLA and ProSHLA utilize the network structure while approximating the objective function. The L-shaped and inexact cut algorithms are time consuming because 2000 samples are used in this experiment, corresponding to a large number of cuts for the L-shaped and

Table 2: **Computational time for SHLA and ProSHLA.**

| | | Computation time (s) | | | | |
|---|---|---|---|---|---|---|
| $\lvert\mathcal{L}\rvert$ | $N_R$ | L-shaped | Inexact cut | SHLA-1 | SHLA-2 | ProSHLA |
| 5 | 400 | 153 | 76 | 28 | 28 | 43 |
| 10 | 800 | 535 | 382 | 90 | 95 | 148 |
| 15 | 1200 | 1140 | 598 | 224 | 217 | 332 |
| 20 | 1600 | 2277 | 1084 | 417 | 420 | 628 |
| 25 | 2000 | 4190 | 1843 | 676 | 790 | 1107 |
| 30 | 2400 | 5491 | 2338 | 1112 | 1066 | 1559 |
| 35 | 2800 | 8075 | 4249 | 1539 | 1531 | 2341 |
| 40 | 3200 | 18636 | 8154 | 3010 | 3428 | 5307 |

inexact cut algorithms. The experiment results also demonstrate that the inexact cut algorithm requires less computation time than the L-shaped algorithm because there exist fewer valid cuts in the inexact cut algorithm than the optimality cuts in the L-shaped algorithm. Furthermore, the computation time of SHLA ($\delta = 1$) is nearly the same as that of SHLA ($\delta = 2$). This finding indicates that the choice of $\delta$ does not affect the computation time. However, ProSHLA ($\delta = 2$) requires more computation time than SHLA ($\delta = 2$) because the projection steps in ProSHLA are time consuming. In the following subsections, we focus on the convergence performance of SHLA and ProSHLA. According to the results in Table 1, the solutions of the L-shaped algorithm are very close to those of PB. Thus, we only demonstrate the results of PB, the myopic algorithm, SHLA, and ProSHLA. Here, the solutions for PB and the myopic algorithm are used as the lower and upper bounds, respectively.

## *4.3. Convergence performance*

We conduct a set of experiments to examine the convergence rate of ProSHLA and SHLA. In this part, we select the second instance ($N_L = 10$ and $N_R = 800$) for the purpose of illustration. The number of samples is set from 20 to 640, and the results of each iteration are recorded. As shown in Figure 3, ProSHLA and SHLA-1 exhibit a remarkably high convergence rate.

To further examine the manner in which $\delta$ affects the convergence performance, we conduct a set of experiments in which we increase the number of samples from 20 to 640 and $\delta$ from 1 to 16. For each combination, we record the sample means of PB, the solutions of ProSHLA and SHLA, and the myopic method. Figures 4 and 5 show the 3D plots of the results. In Figure 4, the

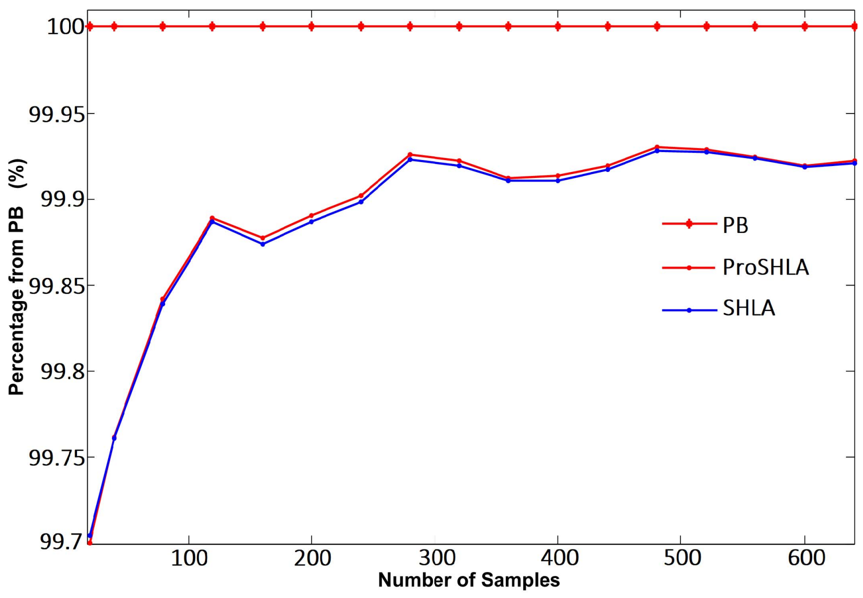

Figure 3: **Convergence rate of ProSHLA and SHLA.**

SHLA and ProSHLA layers are extremely close to the PB layer, implying that SHLA and ProSHLA exhibit a rapid convergence rate for various $\delta$. Furthermore, it demonstrates that the performance of ProSHLA slightly exceeds that of SHLA. To further explore the difference between ProSHLA and SHLA intuitively, we illustrate SHLA and ProSHLA separately (without PB and the myopic algorithm) in Figure 5. As shown in the figure, the performance of SHLA and ProSHLA is affected by $\delta$, and a small $\delta$ leads to good solutions.

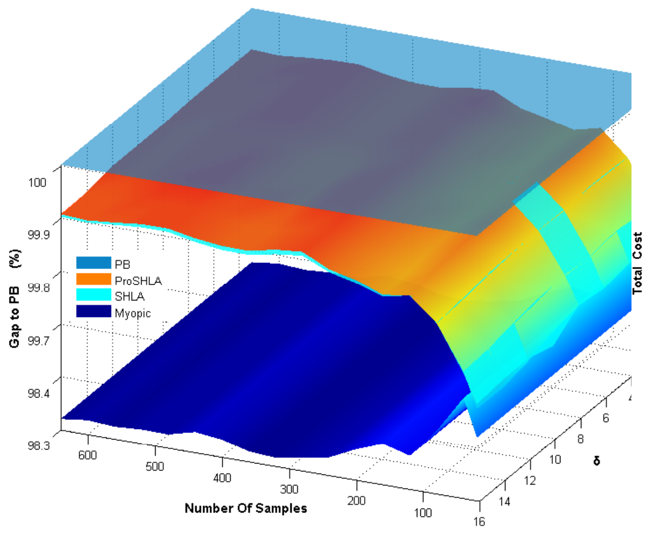

Figure 4: **Gaps to PB for various $\delta$.**

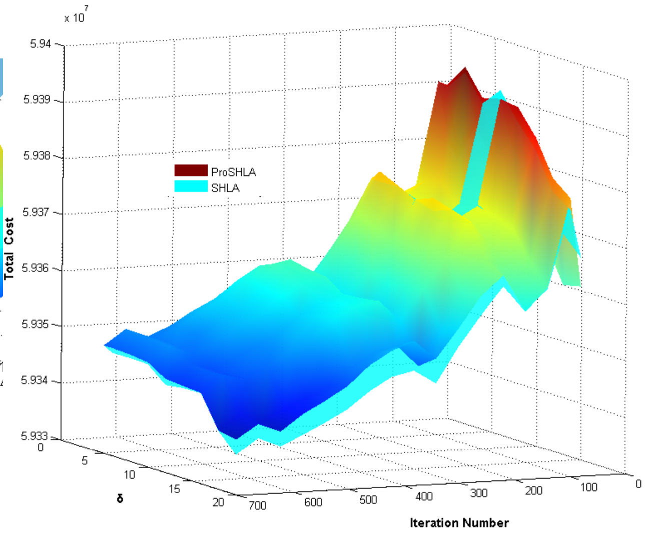

Figure 5: **Comparison of ProSHLA and SHLA for various $\delta$.**

Table 3 below provides further details of the convergence performance of SHLA and ProSHLA for various $\delta$. It also clearly demonstrates that in conjunction with the small $\delta$, the performance of SHLA and ProSHLA is close to that of PB.

Table 3: **Performance under various $\delta$** (number of samples is 2000).

| | Total cost - % gap to PB (computation time in seconds) | | | |
|---|---|---|---|---|
| $\delta$ | PB | Myopic | SHLA | ProSHLA |
| 1 | -59397423 | 1.6251% | 0.0989%(909.5) | 0.0989%(1441.5) |
| 2 | -59397423 | 1.6251% | 0.0997%(907.9) | 0.0995%(1506.5) |
| 4 | -59397423 | 1.6251% | 0.1001%(901.8) | 0.0997%(1503.6) |
| 6 | -59397423 | 1.6251% | 0.1005%(911.5) | 0.1004%(1553.5) |
| 8 | -59397423 | 1.6251% | 0.1028%(901.7) | 0.1024%(1535.6) |
| 16 | -59397423 | 1.6251% | 0.1189%(920.3) | 0.1150%(1565.4) |

## *4.4. Performance under various distributions*

We attempt to examine whether or not the distribution of random demands affects the performance. As the exponential distribution has been examined in Subsections 4.2 and 4.3, we consider another two distributions in this subsection: the normal and uniform distributions. The results are presented in Tables 4 and 5.

Table 4: **Performance under uniform distributions.**

| | Total cost - % from optimal (CPU time in seconds) | | | | |
|---|---|---|---|---|---|
| *No.Samples* | PB | Myopic | SHLA-1 | SHLA-2 | ProSHLA |
| 20 | -47033196 | 7.0596% | 0.6400%(10.1) | 0.6411%(12.0) | 0.6410%(14.5) |
| 40 | -46953805 | 7.2430% | 0.5728%(18.7) | 0.5737%(20.5) | 0.5733%(26.9) |
| 80 | -46924799 | 7.1129% | 0.5377%(37.2) | 0.5388%(40.4) | 0.5382%(54.8) |
| 120 | -46838644 | 7.1220% | 0.4179%(56.1) | 0.4186%(61.6) | 0.4184%(83.0) |
| 160 | -46855277 | 7.1063% | 0.4318%(74.9) | 0.4327%(87.8) | 0.4326%(110.3) |
| 200 | -46854227 | 7.1248% | 0.4129%(93.0) | 0.4140%(114.0) | 0.4131%(138.8) |
| 240 | -46833626 | 7.1516% | 0.4118%(112.4) | 0.4130%(123.2) | 0.4120%(167.1) |
| 280 | -46823765 | 7.1447% | 0.3846%(131.3) | 0.3866%(151.3) | 0.3851%(195.2) |
| 320 | -46833694 | 7.1192% | 0.4010%(149.0) | 0.4026%(170.7) | 0.4012%(221.8) |
| 360 | -46814211 | 7.1201% | 0.3901%(163.8) | 0.3908%(190.7) | 0.3901%(260.2) |
| 400 | -46817694 | 7.1171% | 0.3823%(183.5) | 0.3829%(214.0) | 0.3824%(280.4) |
| 440 | -46810860 | 7.1177% | 0.3826%(200.8) | 0.3831%(237.8) | 0.3826%(309.8) |
| 480 | -46804103 | 7.1203% | 0.3742%(220.6) | 0.3749%(254.0) | 0.3744%(338.6) |
| 520 | -46803204 | 7.1232% | 0.3718%(238.2) | 0.3732%(279.4) | 0.3719%(370.3) |
| 560 | -46802228 | 7.1154% | 0.3766%(257.6) | 0.3787%(296.4) | 0.3766%(399.8) |
| 600 | -46805244 | 7.1204% | 0.3783%(276.6) | 0.3799%(331.6) | 0.3783%(431.9) |
| 640 | -46800538 | 7.1302% | 0.3792%(298.0) | 0.3807%(342.6) | 0.3796%(462.2) |

As shown in Tables 4 and 5, the results of *normal* and *uniform* distributions are consistent with those of *exponential* distribution. In addition, ProSHLA and SHLA achieve near-optimal

Table 5: **Performance under normal distributions.**

| | Total cost - % from optimal (CPU time in seconds) | | | | |
|---|---|---|---|---|---|
| $No.Samples$ | PB | Myopic | SHLA-1 | SHLA-2 | ProSHLA |
| 20 | -47033196 | 6.5779% | 0.0458%(10.4) | 0.0473%(10.6) | 0.0473%(14.3) |
| 40 | -46953805 | 6.6132% | 0.0314%(18.4) | 0.0349%(18.0) | 0.0349%(27.1) |
| 80 | -46924799 | 6.6001% | 0.0225%(36.6) | 0.0259%(36.1) | 0.0256%(54.4) |
| 120 | -46838644 | 6.6022% | 0.0106%(54.9) | 0.0149%(54.3) | 0.0140%(83.1) |
| 160 | -46855277 | 6.6072% | 0.0184%(73.7) | 0.0222%(72.1) | 0.0216%(114.0) |
| 200 | -46854227 | 6.6138% | 0.0165%(92.4) | 0.0201%(89.8) | 0.0196%(137.2) |
| 240 | -46833626 | 6.6135% | 0.0152%(111.0) | 0.0189%(107.3) | 0.0186%(165.1) |
| 280 | -46823765 | 6.6125% | 0.0161%(127.8) | 0.0199%(126.7) | 0.0200%(193.2) |
| 320 | -46833694 | 6.6075% | 0.0178%(146.4) | 0.0214%(144.7) | 0.0218%(218.5) |
| 360 | -46814211 | 6.6074% | 0.0171%(163.8) | 0.0209%(160.9) | 0.0206%(246.7) |
| 400 | -46817694 | 6.6078% | 0.0173%(182.9) | 0.0208%(179.5) | 0.0209%(276.7) |
| 440 | -46810860 | 6.6084% | 0.0177%(204.3) | 0.0215%(198.5) | 0.0213%(302.7) |
| 480 | -46804103 | 6.6083% | 0.0173%(223.2) | 0.0210%(217.1) | 0.0208%(332.9) |
| 520 | -46803204 | 6.6093% | 0.0186%(239.5) | 0.0221%(233.6) | 0.0219%(362.2) |
| 560 | -46802228 | 6.6091% | 0.0187%(264.4) | 0.0219%(251.5) | 0.0221%(390.4) |
| 600 | -46805244 | 6.6092% | 0.0177%(286.8) | 0.0210%(271.2) | 0.0209%(421.2) |
| 640 | -46800538 | 6.6101% | 0.0175%(309.8) | 0.0207%(291.2) | 0.0208%(488.3) |

solutions and exhibit rapid convergence rate. The results are consistent with the fact that the algorithms provided do not require any information on the demand distribution, that is, they are distribution-free.

### *4.5. Performance on a high-dimensional resource allocation problem*

In this subsection, we evaluate the performance of ProSHLA on a high-dimensional resource allocation problem. The problem can be illustrated as follows. There is a set of production facilities (with warehouses) $\mathcal{L}$ and a set of retailers $\mathcal{R}$. First, an amount $x_{ij}$ is transported from the production facility $i$ to a warehouse or retailer location $j$ prior to retail demand realization in Stage 1. When the retail demands are known, we then move $y_{ij}$ products from location $i$ to retailer location $j$. In addition, there exist various types of demands at each retailer location $j$, indexed by $t \in \mathcal{T}$: $D_j^t$ is the demand of type $t$ at retailer location $j$. We provide $p_i^t$ units of demand of type $t$ at retailer location $i$, and the production capacity of facility $i$ is denoted as $cap_i$.

Subsequently, we can formulate the problem as follows:

$$\min \sum_{i\in\mathcal{L}} \sum_{j\in\mathcal{L}\cup\mathcal{R}} c^1_{ij} x_{ij} + E_\omega[Q(x,\omega)], \tag{52}$$

subject to

$$\begin{aligned}
\sum_{j\in\mathcal{L}\cup\mathcal{R}} x_{ij} &\leq cap_i, \quad \forall i \in \mathcal{L}, && (53)\\
\sum_{i\in\mathcal{L}} x_{ij} &= s_j, \quad \forall j \in \mathcal{L}\cup\mathcal{R}, && (54)\\
x_{ij},\ s_j &\geq 0, \quad \forall i \in \mathcal{L},\ \forall j \in \mathcal{L}\cup\mathcal{R}, && (55)
\end{aligned}$$

where the recourse function $Q(x,\omega)$ is given as follows:

$$Q(x,\omega) = \min \sum_{i\in\mathcal{L}\cup\mathcal{R}} \sum_{j\in\mathcal{R}} c^2_{ij} y_{ij} - \sum_{i\in\mathcal{R}} \sum_{t\in\mathcal{T}} r^t_i p^t_i, \tag{56}$$

subject to

$$\begin{aligned}
\sum_{j\in\mathcal{R}} y_{ij} &= s_i, \quad \forall i \in \mathcal{L}\cup\mathcal{R}, && (57)\\
\sum_{i\in\mathcal{L}\cup\mathcal{R}} y_{ij} &= \sum_{t\in\mathcal{T}} p^t_j, \quad \forall j \in \mathcal{R}, && (58)\\
p^t_j &\leq D^t_j(\omega), \quad \forall t \in \mathcal{T},\ j \in \mathcal{R}, && (59)\\
y_{ij},\ p^t_j &\geq 0, \quad \forall i \in \mathcal{L}\cup\mathcal{R},\ j \in \mathcal{R},\ t \in \mathcal{T}. && (60)
\end{aligned}$$

In the first stage, we set $c^1_{ij} = c^1_0 + c^1_1 d_{ij}$, where $d_{ij}$ is the Euclidean distance between locations $i$ and $j$, $c^1_0$ is the production cost for each product, and $c^1_1$ is the transportation cost per mile. For the second-stage costs, we set

$$c^2_{ij} = \begin{cases} c^2_1 d_{ij} & \text{if } i \in \mathcal{L} \text{ or } i = j, \\ c^2_0 + c^2_1 d_{ij} & \text{if } i \in \mathcal{R} \text{ and } i \neq j. \end{cases}$$

$c^2_0$ to represent the fixed charge for moving each product from one retailer location to another, and $c^2_1$ is the transportation cost per mile in the second-stage. We associate revenue $r^t_i$ with one unit of the demand type $t$ that occurs in retailer location $i$. Our problem instances differ in the number of products and $|\mathcal{L}\cup\mathcal{R}|$, which determines the dimensionality of the recourse function. Note that the recourse function $Q(x,\omega)$ for this resource allocation problem is nonseparable.

Similarly, we use the L-shaped algorithm (Van Slyke and Wets, 1969) and the inexact cut algorithm (Zakeri et al., 2000) as benchmarks; these two algorithms are Benders decomposition-based stochastic programming methods. As the speed of convergence is quite important in practical problems, our focus is on the convergence rate. To measure the convergence rate of different methods, we implement each algorithm for 40, 160, 640, 4000, and 8000 iterations and make a side-by-side comparison of the algorithms when the number of iterations increases. For ProSHLA ($\delta = 2$), the number of iterations refers to the number of demand samples used. For the L-shaped and inexact cut algorithms, the number of iterations refers to the number of cuts used to approximate the expected recourse function.

The experiment results are shown in Table 6. For all problem instances, we use the L-shaped algorithm to find the optimal solution. The numbers in the table represent the percent deviation between the optimal and objective values that correspond to the solution obtained after a certain number of iterations. Table 6 also lists the computation time per iteration. We present the experimental results on five problem instances, which vary in the dimensionality of the recourse function.

Table 6: **Percent deviation over optimal solution with different algorithm costs**

| | | | | Number of iterations | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *No.* | $\lvert\mathcal{L} \cup \mathcal{R}\rvert$ | $N_P$ | *Algorithm* | 40 | 160 | 640 | 4000 | 8000 | Sec./iter. |
| | | | ProSHLA | 13.26 | 8.61 | 2.93 | 2.73 | 0.47 | 0.01 |
| 1 | 6 | 10 | L-shaped | 1.17 | 0 | 0 | 0 | 0 | 0.07 |
| | | | Inexact cut | 0.96 | 0 | 0 | 0 | 0 | 0.05 |
| | | | ProSHLA | 10.58 | 3.01 | 0.61 | 0.12 | 0.06 | 0.07 |
| 2 | 10 | 200 | L-shaped | 1.85 | 0 | 0 | 0 | 0 | 0.26 |
| | | | Inexact cut | 1.31 | 0 | 0 | 0 | 0 | 0.21 |
| | | | ProSHLA | 7.22 | 1.22 | 0.42 | 0.05 | 0.03 | 0.30 |
| 3 | 20 | 400 | L-shaped | 10.46 | 1.16 | 0 | 0 | 0 | 1.13 |
| | | | Inexact cut | 6.63 | 0.98 | 0 | 0 | 0 | 1.04 |
| | | | ProSHLA | 6.03 | 0.82 | 0.34 | 0.02 | 0.01 | 0.83 |
| 4 | 40 | 800 | L-shaped | 23.57 | 3.23 | 0.31 | 0 | 0 | 9.53 |
| | | | Inexact cut | 17.16 | 2.24 | 0.13 | 0 | 0 | 8.72 |
| | | | ProSHLA | 5.68 | 0.78 | 0.15 | 0 | 0 | 2.68 |
| 5 | 100* | 2000 | L-shaped | 44.84 | 14.51 | 1.38 | 0.03 | 0.02 | 36.53 |
| | | | Inexact cut | 29.56 | 8.14 | 0.91 | 0.03 | 0.02 | 30.98 |

*Note* : Figures represent the deviation from the best objective value known. *Optimal solution not found.

As shown in Table 6,Column 1 shows the index number for the problem instance. Column 2 shows the number of locations while Column 3 presents the number of products. Column 4 lists the methods used in the experiment. Columns 5 to 9 provide the percentage deviation from the optimal value. Column 10 presents the computation time per iteration. The results demonstrate that ProSHLA can produce high quality solutions rather quickly for large problems and achieve consistent performance under different problem instances (Instances 2-5). In particular, when the dimensionality of the problem is increased, ProSHLA continues to provide consistent performance, and its computation time per iteration is short. This feature makes ProSHLA appealing for large-scale problems. Compared with the two Benders decomposition-based algorithms, ProSHLA is more competitive for large problems because separable approximations scale much more readily to high-dimensional problems. In the first problem instance, when the network flow values are small and the network size is also small, due to approximation errors introduced by the separable approximation, solutions produced by our method may have a slow rate of convergence to an optimal solution (with a percent deviation gap around 2.73% after 4000 iterations, and requiring 8000 iterations to reach a percent deviation gap of 0.47%).

According to the above results, ProSHLA is a promising method for two-stage stochastic programs, and solutions produced by ProSHLA always guarantee to converge to an optimal solution even when approximating a nonseparable recourse function. Due to its simplicity and fast runtime, ProSHLA is an attractive candidate for problems where the cost of one experiment is high, and can be used as an initialization routine for two-stage stochastic programming methods to achieve high-quality feasible solutions.

## 5. Conclusion

In this study, we propose a new learning algorithm that is a hybrid of piecewise linear approximation and stochastic subgradient methods. The algorithm is proven as convergent for general two-stage stochastic programs. Furthermore, the results of this work indicate that with respect to two-stage stochastic network recourse problems, a pure piecewise linear approximation algorithm is convergent when the initial piecewise linear functions are sufficiently *fine*. The numerical experiments indicate that both algorithms are efficient and exhibit a fast convergence rate. Furthermore, the computation time appears to be indifferent to various values of granularity ($\delta$), and a small $\delta$ of the initial function leads to a high convergence rate. Finally, we also demonstrate that the algorithms are independent of the distributions of randomness and competitive for high-dimensional problems. This study does not explore whether the method can be extended to other applications, especially to general two-stage stochastic integer programs. This topic should be investigated in the future, and we expect that numerous other applications can be solved by this approach.

**Acknowledgments**

This research is partially supported by the National Nature Science Foundation of China (Project No. 71701221 and No. 71431007), the MOE (Ministry of Education in China) Project of Humanities and Social Sciences (Project No. 17YJ630235) and the Fundamental Research Funds for the Central Universities (Project No. 17wkpy24). We would also like to thank two anonymous referees for constructive comments which helped to improve the paper.

## References

Alem, D., Clark, A., & Moreno, A. (2016). Stochastic network models for logistics planning in disaster relief. *European Journal of Operational Research*, **255**(1), 187–206.

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

Bouzaiene-Ayari, B., Cheng, C., Das, S., Fiorillo, R., & Powell, W.B. (2016). From single commodity to multiattribute models for locomotive optimization: A comparison of optimal integer programming and approximate dynamic programming. *Transportation Science*, **50**, 366–389.

Cheung, R.K., & Chen, C.Y. (1998). A two-stage stochastic network model and solution methods for the dynamic empty container allocation problem. *Transportation Science*, **32**(2), 142–162.

Cheung, R.K., & Powell, W.B. (1996). An algorithm for multistage dynamic networks with random arc capacities, with an application to dynamic fleet management. *Operations Research*, 44(6),951-963.

Cheung, R.K., & Powell, W.B. (2000). SHAPE–a stochastic hybrid approximation procedure for two-stage stochastic programs. *Operations Research*, **48**(1), 73–79.

Cordeau, J-F., Soumis, F., & Desrosiers, J. (2000). A Benders decomposition approach for the locomotive and car assignment problem. *Transportation Science*, **34**(2), 133–149.

Ermoliev, Y. (1988). Stochastic quasigradient methods. *In Numerical Techniques for Stochastic Optimization*, Springer-Verlag, New York.

Girardeau, P., Leclere, V., & Philpott, A. B. (2015). On the convergence of decomposition methods for multistage stochastic convex programs. *Mathematics of Operations Research*, **40**(1), 130–145.

Godfrey, G.A., & Powell, W.B. (2002). An adaptive dynamic programming algorithm for dynamic fleet management i: single period travel times. *Transportation Science*, **36**(1), 21–39.

Guigues, V. (2014). SDDP for some interstage dependent risk-averse problems and application to hydro-thermal planning. *Computational Optimization and Applications*, **57**, 167–203.

Guigues, V. (2016). Convergence analysis of sampling-based decomposition methods for risk-averse multistage stochastic convex programs. *SIAM Journal on Optimization*, **26**, 2468–2494.

Guigues, V. (2017a). Inexact cuts for deterministic and stochastic dual dynamic programming applied to convex nonlinear optimization problems. *arXiv preprint arXiv:1707.00812* (2017).

Guigues, V. (2017b). Multistep stochastic mirror descent for risk-averse convex stochastic programs based on extended polyhedral risk measures. *Mathematical Programming*, **163**, 169–212.

Guigues, V. (2018). Inexact cuts in deterministic and stochastic dual dynamic programming applied to linear optimization problems. *arXiv preprint arXiv:1801.04243* (2018).

Guigues, V., & Römisch, W. (2012). Sampling-based decomposition methods for multistage s-tochastic programs based on extended polyhedral risk measure. *SIAM Journal on Optimization*, **22**, 286–312.

Higle, J.L., & Sen, S. (1991). Stochastic decomposition: an algorithm for two-stage linear programs with recourse. *European Journal of Operational Research*, **16**(3), 650–669.

Infanger, G., & Morton, D. (1996). Cut sharing for multistage stochastic linear programs with interstage dependency. *Mathematical Programming*, **75**, 241–256.

Kim, K., & Mehrotra, S. (2015). A two-stage stochastic integer programming approach to integrated staffing and scheduling with application to nurse management. *Operations Research*, **63**, 1431–1451.

Kleywegt, A.J., Shapiro, A., & Homem de Mello T. (2001). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, **12**(2), 479–502.

Lohmann, T., Hering, A.S, & Rebennack, S. (2016). Spatio-temporal hydro-inflow forecasting of multireservoir inflows for hydro-thermal scheduling. *European Journal of Operational Research*, **255**(1), 243-258.

Löhndorf, N., Wozabal, D., & Minner, S. (2013). Optimizing trading decisions for hydro storage systems using approximate dual dynamic programming. *Operations Research*, **61**(4), 1182–1213.

Long, Y., Lee, L.H., & Chew, E.P. (2012). The sample average approximation method for empty container repositioning with uncertainties. *European Journal of Operational Research* , **222**(1), 65–75.

Moreno, A., Alem, D., Ferreira, D., & Clark, A. (2018). An effective two-stage stochastic multi-trip location-transportation model with social concerns in relief supply chains. *European Journal of Operational Research*, **269**(3), 1050–1071.

Nascimento, J., & Powell, W.B. (2013). An optimal approximate dynamic programming algorithm for concave, scalar storage problems with vector-valued controls. *IEEE Transactions on Automatic Control*, **58**(12), 2995–3010.

Nemirovski, A., Juditsky, A., Lan, G., & Shapiro, A. (2009). Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, **19**(4), 1574–1609.

Neveu, J. (1975). Discrete parameter martingales. North Holland, Amsterdam.

Pereira, M.V.F., & Pinto, L.M.V.G. (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, **52**, 359–375.

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

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

Polyak, B.T., & Juditsky, A.B. (1992). Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, **30**(4), 838–855.

Powell, W.B., Ruszczyński, A., & Togaloglu, H. (2004). Learning algorithms for separable approximation of discrete stochastic optimization problems. *Mathematics of Operations Research*, **29**(4), 814–836.

Rebennack, S. (2016). Combining sampling-based and scenario-based nested benders decomposition methods: application to stochastic dual dynamic programming. *Mathematical Programming*, **156**(1), 343–389.

Restrepo, M.I., Gendron, B., & Rousseau, L.M. (2017). A two-stage stochastic programming approach for multi-activity tour scheduling. *European Journal of Operational Research*, **262**(2), 620–635.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, **22**(3), 400–407.

Rockafellar, R.T., & Wets, J.B. (1988). A note about projections in the implementation of stochastic quasigradient methods. *Numerical Techniques for Stochastic Optimization*, *Springer Ser.Comput.Math*, vol. 10. Springer, Berlin, 385–392.

Ruszczyński, A. (1987). A linearization method for nonsmooth stochastic optimization problems. *Mathematics of Operations Research*, **12**, 32–49.

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

Shapiro, A., Dentcheva, D., & Ruszczyński, A. (2014). Lectures on stochastic programming:

modeling and theory, second edition. *Society for Industrial and Applied Mathematics*, Philadelphia.

Shapiro, A., Tekaya, W., da Costa, J., & Soares, M. (2013). Risk neutral and risk averse stochastic dual dynamic programming method. *European Journal of Operational Research*, **224**(1), 375–391.

Song, D.P., & Dong, J.X. (2008). Empty container management in cyclic shipping routes. *Maritime Economics & Logistics*, **10**(4), 335–361.

Steeger, G., Lonhmann, T., & Rebennack, S. (2018). Strategic bidding for a price-maker hydroelectric producer: stochastic dual dynamic programming and lagrangian relaxation. *IISE Transactions*, **50**(11), 929–942.

Taylor, H.M. (1990). Martingales and random walks. Volume 2, Elsevier Science Publishers B.V.

Van Slyke, R.M., & Wets, R.J-B. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, **17**(4), 638–663.

Wallace, S.W., & Ziemba, W.T. (2005). *Applications of Stochastic Programming.* MOS-SIAM Series on Optimization 5.

Zakeri, G., Philpott, A.B., & Ryan, D.M. (2000). Inexact cuts in Benders decomposition. *SIAM Journal on Optimization*, **10**(4), 643–657.