Chapter

Approximate Dynamic Programming: An Efficient Machine Learning Algorithm

Zhou Shaorui, Cai Ming and Zhuo Xiaopo

Abstract

We propose an efficient machine learning algorithm for two-stage stochastic programs. This machine learning algorithm is termed as projected stochastic hybrid learning algorithm, and consists of stochastic sub-gradient and piecewise linear approximation methods. We use the stochastic sub-gradient and sample information to update the piecewise linear approximation on the objective function. Then we introduce a projection step, which implemented the sub-gradient methods, to jump out from a local optimum, so that we can achieve a global optimum. By the innovative projection step, we show the convergent property of the algorithm for general two-stage stochastic programs. Furthermore, for the network recourse problem, our algorithm can drop the projection steps, but still maintains the convergence property. Thus, if we properly construct the initial piecewise linear functions, the pure piecewise linear approximation method is convergent for general two-stage stochastic programs. The proposed approximate dynamic programming algorithm overcomes the high dimensional state variables using methods from machine learning, and its logic capture the critical ability of the network structure to anticipate the impact of decisions now on the future. The optimization framework, which is carefully calibrated against historical performance, make it possible to introduce changes in the decisions and capture the collective intelligence of the experienced decisions. Computational results indicate that the algorithm exhibits rapid convergence.

Keywords: stochastic programming, piecewise linear approximation, machine learning, network, approximate dynamic programming

1. Introduction

Optimal learning addresses the challenge of how to collect information, as efficiently as possible, to make a decision in the present such that it minimizes the expectation of costs in the future with uncertainty. Collecting information is usually time consuming and expensive. For example, several large shippers, such as Amazon, Walmart, and IKEA, need to decide the quantity of products to ship from plants to warehouses to satisfy the retailers' demand. The retailer usually makes their decisions before knowing the real demand. Then, after they know the retail demand, they optimize the shipping plans between retailers and warehouses. The aforementioned problems generally can be treated as the two-stage stochastic programs. The decisions that the retailer made now (Stage 1) will determines the state while solving the problem in future (Stage 2). Therefore, an optimal decision can be made now if we can compute the expected cost function (the recourse function) of Stage 2. In this chapter, we propose an efficient machine learning algorithm that can collect information very efficiently based on the knowledge gradient and solve the problem optimally. The main gap between MAT and proposed algorithm is that our proposed can collect the information based on knowledge gradient and overcomes the "curse of dimensionality". Besides, it can transfer the problem into a polynomial solvable problem and has been proven convergent theoretically.

1.1 Motivation

Optimal learning is a rich filed that includes contributions from different communities. At the moment, this chapter focus on optimal learning in two-stage stochastic program, which is a practically important problem. Problems of this type arise in several areas in dynamic programming, in which the decision maker need to make temporal and spatial decisions before realizing events that will influence the decisions. For example, in empty container repositioning problems [1], shipping companies need to reposition empty containers before realizing the demand. In locomotive planning problems [2], railroads have to decide the schedule of trains in which locomotives are assigned before disruptions occur across the railway network. For relief distribution problems [3], humanitarian decision makers need to distribute emergency aid to disaster locations when the emergency aid materials are very scarce amidst great uncertainties. For job scheduling problems [4], the managers need to decide initial staffing levels and their working plans before the demand are realized. Most of the aforementioned applications are fully sequential problems, and they can be modeled as two-stage stochastic programming problems. Hence, the research of two-stage stochastic optimization in this chapter is very important. However, the main obstacle in most practical problem is that the expected cost function in Stage 2 is quite complex due to uncertainty. In this chapter, we propose a hybrid learning algorithm called *projected stochastic hybrid learning algorithm* (ProSHLA) to approximate the expected recourse function for two-stage stochastic programs. In order to demonstrate the efficiency of the algorithm, we also theoretically prove the convergence of the proposed algorithm mathematically.

In essence, ProSHLA is a hybrid of stochastic sub-gradient and piecewise linear approximation methods. The core of ProSHLA consists of a series of learning steps those provide information for updating the recourse function through a sequence of piecewise linear separable approximations, and a series of the projection steps those can guarantee convergence by implementing the stochastic sub-gradient method. The mathematical analysis and the computational results all demonstrates that when the initial piecewise linear approximation function is properly constructed for two-stage stochastic programs with network recourse, the learning algorithm can drop the projection steps without sacrificing convergence. Moreover, without the projection step, the learning algorithm only consists of a series of learning steps through a sequence of piecewise linear separable approximations, and can solve the practical complex problems very efficiently. Our innovative finding can help the practitioner and the scholar to understand the open problem that has puzzled them for decades: why does the piecewise linear approximation method can be efficient and convergent for stochastic programs with network recourse in practice. In this chapter, we provide the first theoretical support by our analytical results for the use of the piecewise linear approximation method in solving practical problems.

1.2 Literature review

In this chapter, we consider the two-stage stochastic programming problem as follows:

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

s.t.,

$$Ax = b,$$
$$x \geq 0,$$

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

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

s.t.,

$$W(\omega)y = h(\omega) - T(\omega)x,$$
$$y(\omega) \geq 0(\omega).$$

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

Stochastic programming models and solution methods has been examined by many researchers. Comprehensive reviews and discussions were performed by Wallace and Ziemba [5]. The expected recourse function is extremely complex to evaluate except for a few special cases. There are various approximation methods those can be categorized into four groups. Let $\widehat{Q}(x)$ denote the approximate function. The first group includes scenario methods which use the sample average of $Q(x, \omega_i)$ for several samples, $\omega_1, \omega_2, \dots \omega_N$, to approximate the expected recourse function [6]. The approximation function is usually successively updated by the following function:

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

Generally, the scenario method is very efficient, but it cannot guarantee to obtain the convergent solution.

The second group consists stochastic gradient techniques [7, 8], which updates solutions by using stochastic sub-gradients as directions. Usually, the approximate function can be successively updated by the following function:

$$\widehat{Q}(x) = \left(\overline{g}^k\right)^T x \tag{3}$$

where $\bar{g}^k$ denotes a smoothed estimate of the gradient of the expected recourse function at x for iteration k. This method can be proven convergent by projection [9] or recursive linearization [10], although the drawback of this method is that it is time-consuming.

The third group mainly consists of primal and dual decomposition methods. The use of primal and dual decomposition methods dates back to Benders decomposition [11]. Van Slyke and Wets [12] first adopted the L-shaped algorithm into the application of Benders decomposition to two-stage stochastic programs. Pereira and Pinto [13] proposed the stochastic dual dynamic programming (SDDP) method, which has been widely applied in many areas. SDDP uses Benders cuts to compute an outer approximation of a (convex) recourse function, and constructs feasible dynamic programming policies. SDDP has led to numerous related approximation methods those are based on the same logic but seek to improve the approximation procedures by exploiting the underlying structure of the particular applications. These methods consist of use of inexact cuts [14], risk-averse variants [15], embedding SDDP in the scenario tree framework [16]. The convergence of SDDP and related methods has been proven by [17], for linear programs by Girardeau et al. [18].

The fourth group includes separable approximation methods [19, 20]. This type of methods usually replaces the expected recourse function in Eq. (1) with separable approximation functions as follows:

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

If the separable functions $\widehat{Q}_i(x)$ are piecewise linear or linear, we can replace the expected recourse function in Eq. (1) with $\widehat{Q}(x)$. Then we can solve the problem as a pure network flow problem for network recourse problems, which is polynomial solvable. Thus, it is very efficient. For example, Godfrey and Powell [21] proposed an adaptive piecewise concave approximation (CAVE) algorithm, and the experimental performance of the algorithm shows exceptionally good. However, there was none provable convergent results in their study. In order to provide convergent solutions, Cheung and Powell [19] proposed an approximation algorithm (SHAPE), which uses sequences of strongly convex approximation functions. However, the strongly convex functions require to construct a nonlinear term, and the strongly convex term might damage the pure network structure and need additional computational effort. This chapter intends to introduce an accurate and efficient approximations with the convergence property.

1.3 Contributions of the algorithms

In this chapter, we aim to develop a convergent method that can efficiently approximate the expected recourse function for two-stage stochastic programs. The main contributions are listed as following:

1. We propose a new convergent hybrid learning algorithm to approximate the expected recourse function for two-stage stochastic programs.

2. Through rigorous mathematical analysis, we prove the convergence of the proposed algorithm for general two-stage stochastic programs. The

computational results and mathematical analysis both reveals that the algorithm can drop the projection step without sacrificing the convergence for two-stage stochastic programs with network recourse if we can properly construct the initial piecewise linear approximation functions. That means that a pure piecewise linear approximation can be indeed convergent, which is highly consistent with industry practices. This interesting finding answers the open question which has puzzled scholars for more than a decade: why does the piecewise linear approximation work well for two-stage stochastic programs in industry? Our mathematical analysis can provide the first theoretical support.

3. A series of performance analysis has been conducted. The computational results reveal the efficiency of the proposed algorithms and the proposed algorithms are distribution-free. Furthermore, the convergence rate can be affected by the granularity of the initial function (δ). Small granularity usually leads to a high convergence rate. Finally, the computational results also show that the proposed algorithm is very competitive for high dimensional problems.

4. Compared with MAT, the proposed algorithm can collect the information based on knowledge gradients and use it to update the recourse function by learning steps. It can overcome the "curse of dimensionality". Moreover, it can transfer the problem into a polynomial solvable problem.

The remainder of this chapter is organized as follows. Section 2 presents the description and convergence analysis of the algorithm for general two-stage stochastic programs. The algorithm (without projection steps) for two-stage stochastic programs with network recourse are shown in Section 3. Section 4 demonstrates computational experiments based on an application of the empty container repositioning problem. Section 5 presents the conclusions and outline directions for future research.

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

In this section, ProSHLA is first introduced. Subsequently, we analyze the convergence of ProSHLA for general two-stage stochastic programs.

2.1 Description of ProSHLA

To present ProSHLA mathematically, we let, at each iteration k,
$\alpha_k =$(possibly random) positive step size;
$\overline{Q}(x) =$expected recourse function, that is, $E_\omega[Q(x,\ \omega)]$;
$\widehat{Q}^k(x) =$a convex differentiable approximation of $\overline{Q}(x)$;
$\widehat{q}^k(x) =$a subgradient of $\widehat{Q}^k(x)$ at x, that is $\widehat{q}^k(x) \in \partial\widehat{Q}^k(x)$;
$\bar{g}^k =$a smoothed estimate of the gradient of $\overline{Q}(x)$ at iteration k;
$g^k =$a stochastic subgradient of $\overline{Q}(x)$ at x^k, that is, $g^k \in \partial Q(x^k,\omega^{k+1})$;
$H_k = \{\omega_1, \omega_2, \dots \omega_N\} =$ the history up to (and including) iteration k.

For a general non-smooth convex function $\widehat{Q}(x)$, its sub-differential can be defined as follows:

$$\partial\widehat{Q}(x) = \left\{\widehat{q}(x) \in \mathfrak{R}^n : \widehat{Q}(y) - \widehat{Q}(x) \geq \widehat{q}(x)^{\mathrm{T}} (y - x)\right\}.$$

We combine Eqs. (1), (3), and (4) to form an approximation at iteration k as follows:

$$min\, c_0^T x + \widehat{Q}^0(x) + \left(\overline{g}^k\right)^T x \tag{5}$$

In this study, we approximate the expected recourse function at iteration k via a convex, differentiable approximation $\widehat{Q}^0(x)$ with a linear correction term $\left(\overline{g}^k\right)^T x$. At each iteration, the linear correction term $\left(\overline{g}^k\right)^T x$ are introduced to improve the initial approximation $\widehat{Q}^0(x)$. Note that here we use a convex initial approximation function $\widehat{Q}^0(x)$, whereas SHAPE uses strongly convex approximation functions. SHAPE will introduce a nonlinear term in the approximation function to maintain the strong convexity property, and it might destroy the pure network flow problem structure and demands additional computational effort. Moreover, we do not calculate $\overline{g}^k$ in the usual manner to obtain stochastic sub-gradients in this study. We use the following form in our model instead:

$$min\, c_0^T x + \widehat{Q}^k(x) + \alpha_k \left(g^k - \widehat{q}^k(x)\right)^T x, \tag{6}$$

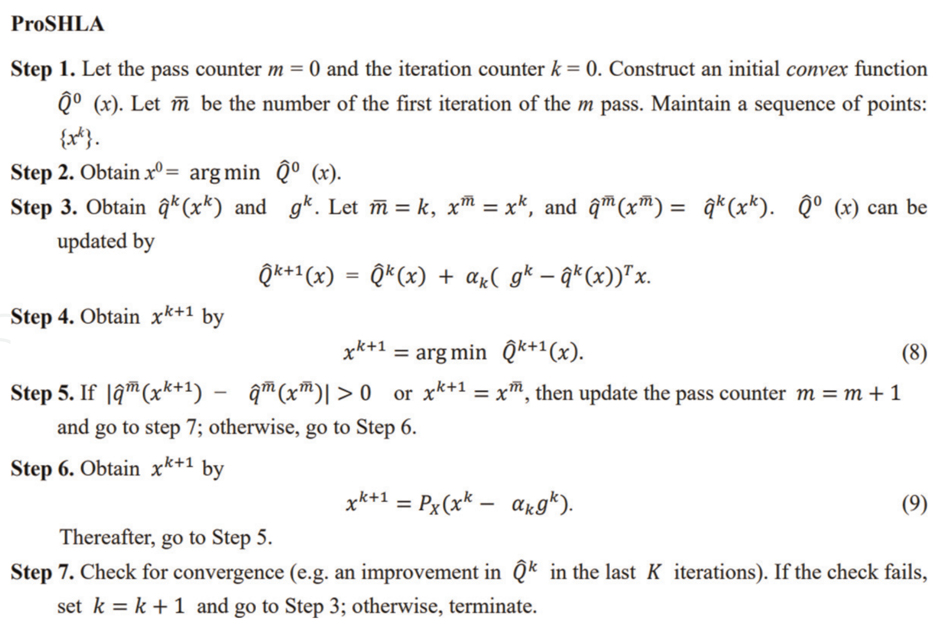
where $\widehat{Q}^k(x)$ is updated as follows:

$$\widehat{Q}^{k+1}(x) = \widehat{Q}^k(x) + \alpha_k \left(g^k - \widehat{q}^k(x)\right)^T x \tag{7}$$

The greatest merit of updating $\widehat{Q}^{k+1}(x)$ in the above way is that it can retain the stochastic sub-gradients $\left(\widehat{q}^k(x^k), \widehat{q}^{k-1}(x^{k-1}), \dots, \widehat{q}^0(x^0)\right)$ used in the previous iterations. Thus, in iteration k, the objective function involves a weighted average of stochastic sub-gradients in the past $(k-1)$ iterations. As shown later in Lemma 2, $\overline{g}^k$ in Eq. (5) is a linear combination of $g^1, g^2, \dots g^{k-1}$.

Let $P_X : R^n \to X$ be the orthogonal projection onto X [9]. Then, we can obtain a sequence of solutions $\{x^k\}$ using the following procedure (**Figure 1**).

Generally, ProSHLA consists of two-level loops. In the first-level loop, there exists a series of *passes*, and in the second-level loop, the exists a series of projection steps, which include the step 5 and 6. We first construct an initial bounded and piecewise linear convex approximation function $\widehat{Q}^0(x)$ at the beginning of the first pass, then the initial solution x^0 can be obtained by solving problem (1). A realization of the random quantity $\omega \in \Omega$ can be drawn, and then we can obtain a stochastic sub-gradient of $\overline{Q}^0(x)$by solving the resulting deterministic problem. Compared with the slope of $\widehat{Q}^0(x)$ and the stochastic sub-gradient at $x = x^0$, the difference of these two slopes can be used as a linear term to update $\widehat{Q}^0(x)$. Subsequently, we can obtain a



ProSHLA

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

Step 2. Obtain $x^0 = \arg\min\ \hat{Q}^0\ (x)$.

Step 3. Obtain $\hat{q}^k(x^k)$ and g^k. Let $\bar{m} = k$, $x^{\bar{m}} = x^k$, and $\hat{q}^{\bar{m}}(x^{\bar{m}}) = \hat{q}^k(x^k)$. $\hat{Q}^0\ (x)$ can be updated by

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

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

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

Step 5. If $|\hat{q}^{\bar{m}}(x^{k+1}) - \hat{q}^{\bar{m}}(x^{\bar{m}})| > 0$ or $x^{k+1} = x^{\bar{m}}$, 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

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

Thereafter, go to Step 5.

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

Figure 1.
The procedure of ProSHLA.

new solution x^{k+1} using the updated approximation function. If the sub-gradient vectors $\hat{q}^{\bar{m}}(x^{k+1})$ of the newly obtained solution x^{k+1} are equal to sub-gradient of solution $x^{\bar{m}}$, that is $\hat{q}^{\bar{m}}(x^{\bar{m}})$, the piecewise linear approximation might have jumped into a local optimum. Subsequently, ProSHLA need to jump out from local optimum by implementing projection steps in the second-level loop. If we obtain a new solution x^{k+1} in the second-level loop and the sub-gradient $\hat{q}^{\bar{m}}(x^{k+1})$ is different from sub-gradient $\hat{q}^{\bar{m}}(x^{\bar{m}})$, then ProSHLA can jump out the second-level loop, and comes to the end of second pass. Thus then, we can repeat the entire process. Finally, ProSHLA will be terminated when the total absolute change in $\widehat{Q}^l(x)$ over a certain number of iterations is low (e.g. $\sum_{l=k-M+1}^{k} \|\widehat{Q}^l(x) - \widehat{Q}^{l-1}(x)| < \delta$).

Here we point out the main difference between SHAPE and ProSHLA. The most remarkable difference is that ProSHLA uses convex approximation functions while SHAPE uses strongly convex approximation functions. The strongly convexity always maintains a nonlinear term in the approximation function. And this term might destroy the pure network flow structure and causes additional computational effort. To overcome the drawback of the SHAPE, we introduce the projection step in the second-level loop and construct approximation functions. Particularly, the approximation functions in the ProSHLA is NOT strictly convex, while it needs to be strictly convex in SHAPE. Without the projection step in the second-level loop, the ProSHLA might stuck in the corner local –optimum for stochastic linear programs. Thus, ProSHLA can work well for most practical stochastic linear programs, because most of practical stochastic programs are piecewise convex problems.

2.2 Convergence analysis of ProSHLA

Firstly, we demonstrate the convergence theorem of ProSHLA in this subsection. Then, several properties of approximation are presented. Finally, we use these properties to prove the convergence of ProSHLA.

Without loss of generality, the following assumptions are listed.

(A.1) $X \subset \Re^n$ is compact and convex.

(A.2) $E_\omega Q(x_1, \omega)$ is convex, finite and continuous on X.

(A.3) g^k is bounded such that $\|g^k\| \le c_1$ for each$\omega \in \Omega$;$\widehat{q}^k$ is bounded such that $\left\|\widehat{q}^k\right\| \le c_2$ for each $\omega \in \Omega$.

(A.4) Piecewise linear function $\widehat{Q}^k(x)$ are convex, implying that

$$\widehat{Q}^k(x_1) - \widehat{Q}^k(y_1) \le \widehat{q}^k(x_1)^T(x_1 - y_1).$$

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

$$0 < \alpha_k < 1, \sum_{k=0}^{\infty} E\{\alpha_k^2\} \le \infty$$

Except for the assumption from (A.1) to (A.5), we also introduce the following assumption to characterize the piecewise linear convex approximation functions.

(A.6) There exists a positive$b$ and a constant δ, such that for any two points $x_1, y_1 \in X$, if$|x_1 - y_1| > \delta$, then $|\widehat{q}(x_1) - \widehat{q}(y_1)| \ge b|x_1 - y_1|$. If there exists $\widehat{q}(x_1)$ and $\widehat{q}(y_1)$ such that $\widehat{q}^k(x_1) - \widehat{q}^k(y_1) = 0$, then$|x_1 - y_1| \le \delta$. If$\delta \to 0$, then the function corresponds to a strongly convex function, if $\delta \to \infty$, then the function becomes purely linear.

Given assumption (A.1)–(A.6), we obtain the following theorem of ProSHLA.

Theorem 1. If assumptions (A.1)–(A.6) are satisfied, then the sequence of $\{x_1^k\}$ generated by algorithm ProSHLA converges almost surely to the optimal solution $x_1^* \in X^*$ of problem (1).

In order to prove the Theorem 1, we need to use the following Martingale convergence theorem and three lemmas.

Martingale Convergence Theorem. A sequence of random variables $\{W^k\}$, which are $\mathcal{H}_k$ measurable, is said to be a super-martingale if there exists the sequence of conditional expectations $E\{\{W^{k+1}|\mathcal{H}_k\}$ and satisfies $E\{\{W^{k+1}|\mathcal{H}_k\} \le W^k$.

Theorem 2. (From reference [22]) Let W^k be a positive super-martingale. Then, W^k converges to a finite random variables a.s.

From the above theorem, we can conclude thatW^k is a stochastic decreasing analogue essentially.

Based on the convexity property, the optimal solution for problem (8) at iteration $\overline{m}$ can be characterized by the following inequality:

$$\left(\widehat{q}^{\overline{m}}\left(x_1^{\overline{m}}\right)\right)^T\left(x_1 - x_1^{\overline{m}}\right) \ge 0, \quad \forall x_1 \in X \tag{8}$$

To obtain Theorem 1, the following three lemmas are required. The first lemma shows that the difference between the solutions of two consecutive update processes will be bounded by the step-size and the stochastic gradient. The second lemma indicates that the approximation $\widehat{Q}^k(x_1)$ is finite. The third lemma shows that T^k (which will be denoted in Lemma 3) is bounded.

Lemma 1. For any two iterations $j \in [\overline{m+1}, \overline{m+2})$, $i \in [\overline{m}, \overline{m+1})$, solutions x_1^j and x_1^i obtained by ProSHLA can be characterized by the following inequality:

$$\alpha_{\overline{m}} g^{\overline{m}} \left(x_1^i - x_1^j\right) \le (\alpha_{\overline{m}} c_1)^2 / b \tag{9}$$

Proof. Consider a special case, where i and j corresponds to two consecutive iterations. Let $i = \overline{m+1} - 1$ and $j = \overline{m+1}$. Based on (10), we can obtain that,

$$\left(\widehat{q}^{\overline{m+1}}\left(x_1^{\overline{m+1}}\right)\right)^T \left(x_1 - x_1^{\overline{m+1}}\right) \ge 0, \forall x_1 \in X \tag{10}$$

According to the approximation function's updating rule, we can conclude that,

$$\left(\widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}}\right) + \alpha_{\overline{m+1}-1}\left(g^{\overline{m+1}-1} - \widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}-1}\right)\right)\right)^T \left(x_1 - x_1^{\overline{m+1}}\right) \ge 0, \forall x_1 \in X \tag{11}$$

Substituting x_1 with $x_1^{\overline{m+1}-1}$ in Eq. (13), we can obtain

$$\begin{aligned} &\alpha_{\overline{m+1}-1}\left(g^{\overline{m+1}-1} - \widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}-1}\right)\right)^T \left(x_1^{\overline{m+1}-1} - x_1^{\overline{m+1}}\right) \\ &\quad \ge \widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}}\right)^T \left(x_1^{\overline{m+1}} - x_1^{\overline{m+1}-1}\right) \end{aligned} \tag{12}$$

If we arrange the above terms, then we can obtain the inequality below:

$$\begin{aligned} &\alpha_{\overline{m+1}-1}\left(g^{\overline{m+1}-1}\right)^T \left(x_1^{\overline{m+1}-1} - x_1^{\overline{m+1}}\right) \\ &\quad \ge \widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}}\right)^T \left(x_1^{\overline{m+1}} - x_1^{\overline{m+1}-1}\right) - \alpha_{\overline{m+1}-1}\left(\widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}-1}\right)\right)^T \left(x_1^{\overline{m+1}} - x_1^{\overline{m+1}-1}\right) \\ &\quad = \left(\widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}}\right) - \widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}-1}\right)\right)^T \left(x_1^{\overline{m+1}} - x_1^{\overline{m+1}-1}\right) \\ &\qquad + \left(1 - \alpha_{\overline{m+1}-1}\right)\left(\widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}-1}\right)\right)^T \left(x_1^{\overline{m+1}} - x_1^{\overline{m+1}-1}\right) \end{aligned} \tag{13}$$

When iteration $\overline{m+1} - 1$ and $\overline{m+1}$ are not in the same update process, then it means that $\widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}-1}\right) \ne \widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}}\right)$. According to assumption (A.6), we can conclude that $\left|\widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}-1}\right) - \widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}}\right)\right| \ge b\left|x_1^{\overline{m+1}-1} - x_1^{\overline{m+1}}\right|$.

According to Eqs. (10) and (13), and $0 < \alpha_{\overline{m+1}-1} < 1$, we can obtain

$$\begin{aligned} &\alpha_{\overline{m+1}-1}\left(g^{\overline{m+1}-1}\right)^T \left(x_1^{\overline{m+1}-1} - x_1^{\overline{m+1}}\right) \\ &\quad \ge b\left\|x_1^{\overline{m+1}-1} - x_1^{\overline{m+1}}\right\|^2 + \left(1 - \alpha_{\overline{m+1}-1}\right)\left(\widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}-1}\right)\right)^T \left(x_1^{\overline{m+1}} - x_1^{\overline{m+1}-1}\right) \\ &\quad \ge b\left\|x_1^{\overline{m+1}-1} - x_1^{\overline{m+1}}\right\|^2. \end{aligned}$$

Applying Schwartz' inequality, we can get the inequality below:

$$\alpha_{\overline{m+1}-1}\left\|g^{\overline{m+1}-1}\right\|\bullet\left\|x_1^{\overline{m+1}-1}-x_1^{\overline{m+1}}\right\|\geq\alpha_{\overline{m+1}-1}\left(g^{\overline{m+1}-1}\right)^T\left(x_1^{\overline{m+1}-1}-x_1^{\overline{m+1}}\right)$$
$$\geq b\left\|x_1^{\overline{m+1}-1}-x_1^{\overline{m+1}}\right\|^2$$

Therefore, $\left\|x_1^{\overline{m+1}-1}-x_1^{\overline{m+1}}\right\|\leq\alpha_{\overline{m+1}-1}\bullet c_1/b$. We can obtain the inequality below:

$$\alpha_{\overline{m+1}-1}\left(g^{\overline{m+1}-1}\right)^T\left(x_1^{\overline{m+1}-1}-x_1^{\overline{m+1}}\right)\leq\left(\alpha_{\overline{m+1}-1}c_1\right)^2/b \tag{14}$$

For any $i\in[\overline{m},\overline{m+1}-1]$, $g^i=g^{\overline{m}}=g^{\overline{m+1}-1}$ and $\widehat{q}^i(x_1)=\widehat{q}^{\overline{m}}(x_1)=\widehat{q}^{\overline{m+1}-1}(x_1)$. Thus,

$$\alpha_{\overline{m}}\left(g^{\overline{m}}\right)^T\left(x_1^i-x_1^{\overline{m+1}}\right)\leq\left(\alpha_{\overline{m}}c_1\right)^2/b \tag{15}$$

For any$i\in[\overline{m},\overline{m+1}-1]$ and$j\in[\overline{m+1},\overline{m+2}-1]$,$g^j=g^{\overline{m+1}}=g^{\overline{m+2}-1}$ for any $x_1\in X$. Thus,

$$\alpha_{\overline{m}}\left(g^{\overline{m}}\right)^T\left(x_1^i-x_1^j\right)\leq\left(\alpha_{\overline{m}}c_1\right)^2/b \tag{16}$$

□

Lemma 2. In iteration k, the approximation function $\widehat{Q}^k(x_1)$ can be written as $\widehat{Q}^k(x_1)=\widehat{Q}^0(x_1)+\left(\overline{g}^k\right)^T x_1$, where $\overline{g}^k$ is a finite vector.

Proof. According to Eq. (5) in Proposition 1, we can conclude that $\overline{g}^{k+1}$ is a linear combination of $g^1,g^2,\dots,g^k$. Since g^k and $\widehat{q}^0(x_1)$ are finite, there will exists a finite and positive vector $\widehat{d}$ such that

$$\widehat{d}\geq\max_k\left|g^k-\widehat{q}^0\left(x_1^k\right)\right| \tag{17}$$

According to Lemma 2 in [19], we can conclude that $\overline{g}^{k+1}\geq\widehat{d}$. □

Let $T^k=\widehat{Q}^k\left(x_1^*\right)-\widehat{Q}^k\left(x_1^k\right)$, where x_1^* represents 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

$$T^j-T^i\leq\alpha_{\overline{m}}\left(g^{\overline{m}}\right)^T\left(x_1^i-x_1^j\right)+\alpha_{\overline{m}}\left(g^{\overline{m}}\right)^T\left(x_1^*-x_1^i\right). \tag{18}$$

Proof. The special case is first considered. Let $i=\overline{m}$ and$j=\overline{m+1}$. By re-writing $x_1^*-x^{\overline{m+1}}$ as$x_1^*-x^{\overline{m}}+x^{\overline{m}}-x^{\overline{m+1}}$, we can obtain the following equation:

$$\widehat{Q}^{\overline{m+1}}(x_1)=\widehat{Q}^{\overline{m+1}-1}(x_1)+\alpha_{\overline{m+1}-1}\left(g^{\overline{m+1}-1}-\widehat{q}^{\overline{m+1}-1}\left(x_1^{\overline{m+1}-1}\right)\right)^T x_1$$
$$=\widehat{Q}^{\overline{m}}(x_1)+\alpha_{\overline{m}}\left(g^{\overline{m}}-\widehat{q}^{\overline{m}}\left(x_1^{\overline{m}}\right)\right)^T x_1$$

Then,

$$\begin{aligned}
T^{\overline{m+1}} - T^{\overline{m}} &= \widehat{Q}^{\overline{m}}(x_1^*) + \alpha_{\overline{m}}\left(g^{\overline{m}} - \widehat{q}^{\overline{m}}(x_1^{\overline{m}})\right)^T x_1^* - \left(\widehat{Q}^{\overline{m}}\left(x_1^{\overline{m+1}}\right)\right. \\
&\quad \left. + \alpha_{\overline{m}}\left(g^{\overline{m}} - \widehat{q}^{\overline{m}}(x_1^{\overline{m}})\right)^T x_1^{\overline{m+1}}\right) - \left(\widehat{Q}^{\overline{m}}(x_1^*) - \widehat{Q}^{\overline{m}}(x_1^{\overline{m}})\right) \\
&= \alpha_{\overline{m}}\left(g^{\overline{m}} - \widehat{q}^{\overline{m}}\right)^T\left(x_1^* - x_1^{\overline{m}} + x_1^{\overline{m}} - x_1^{\overline{m+1}}\right) + \widehat{Q}^{\overline{m}}(x_1^{\overline{m}}) - \widehat{Q}^{\overline{m}}\left(x_1^{\overline{m+1}}\right) \\
&= \underbrace{\left(\widehat{Q}^{\overline{m}}(x_1^{\overline{m}}) - \widehat{Q}^{\overline{m}}\left(x_1^{\overline{m+1}}\right) - \alpha_{\overline{m}}\left(\widehat{q}^{\overline{m}}\right)^T\left(x_1^{\overline{m}} - x_1^{\overline{m+1}}\right)\right)}_{I} \\
&\quad \underbrace{- \alpha_{\overline{m}}\left(\widehat{q}^{\overline{m}}\right)^T(x_1^* - x_1^{\overline{m}})}_{II} + \underbrace{\alpha_{\overline{m}}(g^{\overline{m}})^T\left(x_1^{\overline{m}} - x_1^{\overline{m+1}}\right)}_{III} + \underbrace{\alpha_{\overline{m}}(g^{\overline{m}})^T(x_1^* - x_1^{\overline{m}})}_{IV}
\end{aligned}$$

Considering each part individually, given that $\widehat{q}^{\overline{m}} \in \partial\widehat{Q}^{\overline{m}}(x_1^{\overline{m}})$, by convexity of$\widehat{Q}^{\overline{m}}(x_1^{\overline{m}})$, we can obtain

$$\widehat{Q}^{\overline{m}}(x_1^{\overline{m}}) - \widehat{Q}^{\overline{m}}\left(x_1^{\overline{m+1}}\right) \le \left(\widehat{q}^{\overline{m}}\right)^T\left(x_1^{\overline{m}} - x_1^{\overline{m+1}}\right) \tag{19}$$

Thus, the following expression is applicable.

$$\begin{aligned}
\widehat{Q}^{\overline{m}}(x_1^{\overline{m}}) - \widehat{Q}^{\overline{m}}\left(x_1^{\overline{m+1}}\right) &\le \left(\widehat{q}^{\overline{m}}\right)^T\left(x_1^{\overline{m}} - x_1^{\overline{m+1}}\right) \\
&= (1-\alpha_{\overline{m}})\left(\widehat{q}^{\overline{m}}\right)^T\left(x_1^{\overline{m}} - x_1^{\overline{m+1}}\right) + \alpha_{\overline{m}}\left(\widehat{q}^{\overline{m}}\right)^T\left(x_1^{\overline{m}} - x_1^{\overline{m+1}}\right)
\end{aligned} \tag{20}$$

Given Eq. (10) and $0 < \alpha_{\overline{m}} < 1$, we know that $(I) \le 0$. Additionally, from Eq. (10) and $0 < \alpha_{\overline{m}} < 1$, we know that $(II) \ge 0$.
Thus, $T^{\overline{m+1}} - T^{\overline{m}} \le \alpha_{\overline{m}}(g^{\overline{m}})^T\left(x_1^{\overline{m}} - x_1^{\overline{m+1}}\right) + \alpha_{\overline{m}}(g^{\overline{m}})^T(x_1^* - x_1^{\overline{m}})$.

For any$i \in [\overline{m}, \overline{m+1} - 1]$, $g^i = g^{\overline{m}} = g^{\overline{m+1}-1}$ and $\widehat{q}^i(x_1) = \widehat{q}^{\overline{m}}(x_1) = \widehat{q}^{\overline{m+1}-1}(x_1)$ for any $x_1 \in X$. Therefore,

$$T^{\overline{m+1}} - T^i \le \alpha_{\overline{m}}(g^{\overline{m}})^T\left(x_1^i - x_1^{\overline{m+1}}\right) + \alpha_{\overline{m}}(g^{\overline{m}})^T(x_1^* - x_1^i) \tag{21}$$

For any$i \in [\overline{m}, \overline{m+1} - 1]$ and $j \in [\overline{m+1}, \overline{m+2} - 1]$,$\widehat{Q}^j = \widehat{Q}^{\overline{m+1}} = \widehat{Q}^{\overline{m+2}-1}$ for any $x_1 \in X$. Therefore,

$$T^j - T^i \le \alpha_{\overline{m}}(g^{\overline{m}})^T\left(x_1^i - x_1^j\right) + \alpha_{\overline{m}}(g^{\overline{m}})^T(x_1^* - x_1^i) \tag{22}$$

$\square$

To the proof of Theorem 1, we here consider two scenarios. For the first scenario, ProSHLA does not stop in a given update process. Thus, any update process exhibits finite iterations before the algorithm stops, which means $\overline{m+1} - \overline{m} < M$ for any m (M represents a large number). For the second scenario, ProSHLA might terminate in a given update process. In the following text, Theorem 1 is proven for each scenario.

Scenario 1: ProSHLA does not stop in a given update process.

In the first scenario, a subsequence of $\{x_1^k\}, \{x_1^{\overline{m}}\}$ are considered. We will prove that the subsequence $\{x_1^{\overline{m}}\}$ converges to the true optimal x_1^*. According to the definition of$g^k \in \partial Q(x_1^k, \omega^{k+1})$, we can obtain the following inequality

$$\left(g^k\right)^T \left(x_1^* - x_1^k\right) \le Q\left(x_1^*, \omega^{k+1}\right) - Q\left(x_1^k, \omega^{k+1}\right) \tag{23}$$

where$Q(x_1, \omega^{k+1})$ represents the operational cost function given the outcome ω^{k+1}.

According to Lemma 1, we can obtain the following inquality:

$$\alpha_{\overline{m}}\left(g^{\overline{m}}\right)^T \left(x_1^i - x_1^j\right) \le \left(\alpha_{\overline{m}} c_1\right)^2 / b \tag{24}$$

On the basis of Lemma 3, the difference $T^{\overline{m+1}} - T^{\overline{m}}$ can be described as follows:

$$\begin{aligned} T^{\overline{m+1}} - T^{\overline{m}} &\le \alpha_{\overline{m}}\left(g^{\overline{m}}\right)^T \left(x_1^{\overline{m}} - x_1^{\overline{m+1}}\right) + \alpha_{\overline{m}}\left(g^{\overline{m}}\right)^T \left(x_1^* - x_1^{\overline{m}}\right) \\ &\le -\alpha_{\overline{m}}\left(Q\left(x_1^{\overline{m}}, \ \omega^{\overline{m+1}}\right) - Q\left(x_1^*, \ \omega^{\overline{m+1}}\right)\right) + \alpha_{\overline{m}}\left(g^{\overline{m}}\right)^T \left(x_1^* - x_1^{\overline{m}}\right) \\ &\le -\alpha_{\overline{m}}\left(Q\left(x_1^{\overline{m}}, \ \omega^{\overline{m+1}}\right) - Q\left(x_1^*, \ \omega^{\overline{m+1}}\right)\right) + \left(\alpha_{\overline{m}} c_1\right)^2 / b \end{aligned} \tag{25}$$

Conditional expectation of Eq. (27) with respect to $\mathcal{H}_k$ can be taken on both side and then we can obtain

$$E\left(T^{\overline{m+1}} | \mathcal{H}_{\overline{m}}\right) \le T^{\overline{m}} - \alpha_{\overline{m}}\left(\overline{Q}\left(x_1^{\overline{m}}\right) - \overline{Q}\left(x_1^*\right)\right) + \left(\alpha_{\overline{m}} c_1\right)^2 / b$$

where $\overline{Q}(x_1)$ represents the expected recourse function, that is $E_\omega Q(x_1, \omega)$. Given the conditioning on $\mathcal{H}_k$,$T^{\overline{m}}$,$\alpha_{\overline{m}}$ and $x_1^{\overline{m}}$ on the right-hand side are deterministic. The conditioning $\mathcal{H}_k$ cannot provide any information on $\omega^{\overline{m+1}}$. Hence, we replace $Q\left(x_1, \omega^{\overline{m+1}}\right)$ (for $x_1 = x_1^k$ and $x_1 = x_1^*$) with its expectation $\overline{Q}(x_1)$. Given that $\alpha_{\overline{m}}\left(\overline{Q}\left(x_1^{\overline{m}}\right) - \overline{Q}\left(x_1^*\right)\right) \ge 0$, the sequence

$$W^{\overline{m}} = T^{\overline{m}} + \left(\alpha_{\overline{m}} c_1\right)^2 / b \tag{26}$$

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

$$T^{\overline{m}} \to T^* \quad a.s. \tag{27}$$

We perform the summation of Eq. (27) from 0 to $\overline{M}$ and obtain the following inequality:

$$T^{\overline{M}} - T^0 \le -\sum_{\overline{m}=0}^{\overline{M}} \alpha_{\overline{m}}\left(Q\left(x_1^{\overline{m}}, \ \omega^{\overline{m+1}}\right) - Q\left(x_1^*, \ \omega^{\overline{m+1}}\right)\right) + \sum_{\overline{m}=0}^{\overline{M}} \left(\alpha_{\overline{m}} c_1\right)^2 / b \tag{28}$$

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

$$E\left(T^{\overline{M+1}} - T^0\right)$$

$$\leq -\sum_{\overline{m}=0}^{\overline{M}} E\left\{E\left\{\alpha_{\overline{m}}\left(Q\left(x_1^{\overline{m}}, \quad \omega^{\overline{m+1}}\right)\right)\right\} - Q\left(x_1^*, \quad \omega^{\overline{m+1}}\right)\right\}\mathcal{H}_{\overline{m}}\Big\}\Big\} + E\left\{\sum_{\overline{m}=0}^{\overline{M}} \frac{(\alpha_{\overline{m}} c_1)^2}{b}\right\}$$

$$\leq -\sum_{\overline{m}=0}^{\overline{M}} E\left\{\alpha_{\overline{m}}\left(Q\left(x_1^{\overline{m}}, \quad \omega^{\overline{m+1}}\right) - Q\left(x_1^*, \quad \omega^{\overline{m+1}}\right)\right)|\mathcal{H}_{\overline{m}}\right\} + (c_1)^2/b \sum_{\overline{m}=0}^{\overline{M}} E\left\{\alpha_{\overline{m}}^2\right\}$$

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

$$\sum_{\overline{m}=0}^{\overline{M}} E\left\{\alpha_{\overline{m}}\left(Q\left(x_1^{\overline{m}}, \quad \omega^{\overline{m+1}}\right) - Q\left(x_1^*, \quad \omega^{\overline{m+1}}\right)\right)|\mathcal{H}_{\overline{m}}\right\} < \infty \tag{29}$$

Given that $Q\left(x_1^{\overline{m}}, \omega^{\overline{m+1}}\right) - Q\left(x_1^*, \omega^{\overline{m+1}}\right) \geq 0$ and $\sum_{\overline{m}=0}^{\overline{M}} E\{\alpha_{\overline{m}}^2\} = \infty (a.s.)$, there exists a subsequence $\{\overline{m}\}$ such that

$$\overline{Q}\left(x_1^{\overline{m}}\right) \to \overline{Q}\left(x_1^*\right) \quad a.s.$$

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

$$x_1^{\overline{m}} \to x_1^* \quad a.s.$$

Subsequently, we construct another subsequence $\{x_1^{\overline{m}-1}\}$. Based on Eq. (27),

$$E\left(T^{\overline{M+2}-1} - T^{\overline{M+1}-1}\right) \leq -\sum_{\overline{m}=0}^{\overline{M}} \alpha_{\overline{m}}\left(Q\left(x_1^{\overline{m+1}-1}, \ \omega^{\overline{m+2}-1}\right) - Q\left(x_1^*, \ \omega^{\overline{m+2}-1}\right)\right) + (\alpha_{\overline{m}} c_1)^2/b$$

Like-wise, the following approximation can be proved:

$$x_1^{\overline{m}-1} \to x_1^* \quad a.s.$$

By analogic condition, a very general subsequence $\{x_1^i\}, i \in \left[\overline{m}, \overline{m+1} - 1\right]$ will almost surely converge to x_1^*. Here, we term this type of subsequence X_s.

In the procedure of ProSHLA, the number of all update iterations is finite. Thus, for any subsequence of $\{x_1^k\}$, we can obtain a subsequence that always belongs to X_s. Then, the following conclusion can be obtained:

$$\mathrm{x}_1^{\mathrm{k}} \to \mathrm{x}_1^* \quad \mathrm{a.s.}$$

Scenario 2: ProSHLA halts in a given update process.

For the second scenario, ProSHLA halts in a projection procedure which generates a convergent sequence.

Hence, the conclusion of Theorem 1 can be finally obtained. □.

The above analytical processes demonstrate the convergence property of ProSHLA. According to the above results, we require the function $\widehat{Q}(x)$ to be piecewise linear convex. However, for practitioners are interested in a practically scenarios, in which they usually use separable functions to approximate the expected recourse function for stochastic programs with network recourse. Based on Eq. (4), if the separable functions are piecewise linear or purely linear, then practitioners can easily solve this network recourse problem, because a pure network flow problem is polynomial solvable. In the following section, we will discuss this special practice scenario.

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

In this section, we will discuss the scenario where $\widehat{Q}(x)$ is separable for two-stage stochastic programs with network recourse. For this scenario, we can simplify implement ProSHLA without projection step. We denote this simplified version as the Stochastic Hybrid Learning Algorithm (SHLA), which is described as follows (**Figure 2**).

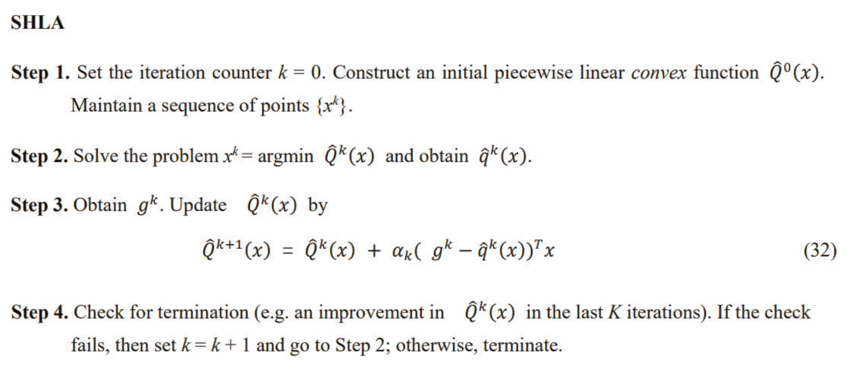
Essentially, SHLA is not convergent. However, if it is applied to two-stage stochastic programs with network recourse, SHLA will enjoys several merits as follows: (1) the solution of $Q(x,\omega)$ is naturally integer; (2) at each iteration, problem $Q(x,\omega)$ is simple network flow problem that can be solved by polynomial algorithm.

Here, if we use separable functions, then assumption (A.6) can be satisfied by the following artificially expression:

$$\widehat{q}^{0}(x_i) < \widehat{q}^{0}(x_i + \delta)$$

Note that for both ProSHLA and SHLA, it allows to choose initial approximation function with different value of δ flexibly. Thus, if δ is set to be 1 for any i, then we can guarantee the following expression:

$$\widehat{q}_i^{0}(x_i) < \widehat{q}_i^{0}(x_i + \delta). \tag{A.7}$$


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 $x^k = \text{argmin } \hat{Q}^k(x)$ and obtain $\hat{q}^k(x)$.

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

$$\hat{Q}^{k+1}(x) = \hat{Q}^k(x) + \alpha_k(g^k - \hat{q}^k(x))^T x \tag{32}$$

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.

Then, we can reach Theorem 3 below.

Theorem 3. If (A.7) is satisfied, SHLA is always convergent for two-stage stochastic programs problem with network recourse.

Proof. For any$x, y \in X$,if there are unequal, we can obtain the following expression according to (A.7).

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

Thus,

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

Hence, if we set $\delta = 1$ and apply ProSHLA for two-stage stochastic programs with network recourse, then ProSHLA can drop the projection step because $\widehat{q}^{k}(x)$ and $\widehat{q}^{k}(y)$ are always unequal. In this situation, ProSHLA is equivalent to SHLA, so SHLA is convergent.

According to the above analysis, we have provided first theoretical convergence support for SHLA-type algorithms which are widely used in numerous applications as mentioned in introduction part. Compared with SHAPE, SHLA does not contain any nonlinear terms so that it can be very efficient. Besides, SHLA can automatically maintain the convexity of the approximation function if the initial piecewise linear functions are properly constructed.

4. Experimental results for performance analysis

In this section, we use two experimental designs to evaluate the performance of the algorithms: (1) An empty container repositioning problem which arises in the context of two-stage stochastic programs with network recourse; and (2) a high dimensional resource allocation problem as an extension experiment. In this section, the empty container repositioning problem is first introduced and then we present the efficiency of ProSHLA and SHLA. Sub-sequentially, we present the convergence of ProSHLA and SHLA, and examine how δ affects convergence performance, and compare the performance under different distributions of random demands. Finally, an extension experiment on a high dimensional resource allocation problem is conducted to evaluate the efficiency our algorithms.

4.1 Problem generator for the empty container repositioning problem

In this subsection, we test our algorithms in an empty container repositioning problem faced by a major Chinese freight forwarder, who need to manage their numerous empty container in a port network which is located in Pearl River Delta in a fixed route from [23]. The port network contains several hubs (large ports) and spokes (small ports). And the demand of empty container is usually uncertain. When the forwarder need to decide the quantity of empty container to ship from one port to another, they did not know the exact demand of container in the future [24, 25]. Thus, we can formulate the problem as a two-stage stochastic programs with network recourse. Before we formally introduce the problem, we present the following notations.

L	=	set of ports;
d_{ij}	=	demand from port i to port$j$ in stage 1;
D_{ij}	=	demand from port i to port$j$ in stage 2;
s_i	=	initial number of empty containers at port i;
S_i	=	number of empty containers at port i in stage 2;
c_{ij}	=	cost for moving an empty container from port i to port$j$;
r_{ij}	=	profit for moving a laden container from port i to port$j$;
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;

Then, the problem can be formulated as follows:

$$min \sum_{i \in L} \sum_{j \in L} \left\{ -r_{ij}x_{ij} + c_{ij}y_{ij} \right\} + E_{\omega}[Q(x, \ \omega)], \tag{30}$$

s.t.,

$$\sum_{j \in L} \left\{ x_{ij} + y_{ij} \right\} = s_i, \qquad \forall i \in L \tag{31}$$

$$\sum_{i \in L} \left\{ x_{ij} + y_{ij} \right\} = s_j, \qquad \forall j \in L \tag{32}$$

$$y_{ij} \geq 0, \ \ \forall i,j \in L \tag{33}$$

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

$$Q(x, \omega) = \ min \sum_{i \in L} \sum_{j \in L} \left\{ -r_{ij}x_{ij}(\omega) + c_{ij}y_{ij}(\omega) \right\} \tag{34}$$

s.t.,

$$\sum_{j \in L} \left\{ x_{ij}(\omega) + y_{ij}(\omega) \right\} = s_i, \qquad \forall i \in L \tag{35}$$

$$\sum_{i \in L} \left\{ x_{ij}(\omega) + y_{ij}(\omega) \right\} = s_j, \qquad \forall j \in L \tag{36}$$

$$y_{ij}(\omega) \geq 0, \ \ \forall i,j \in L \tag{37}$$

In order to evaluate the algorithm, a set of problem instances are created. In this study, the problem generator creates ports in 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. We set the holding cost for a demand to 15 cents per time instance. We set the net profit for a demand to 500 cents per mile. The empty cost is set to 40 cents. The demand D_{ij} between locations i and j is set as follows:

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

where
out_j = outbound potential for port j;
in_i = inbound potential for port i;
v = random variable.

The outbound and inbound potentials for each port represent the capability of the location to generate outbound demand or attract inbound containers. In the generator, We draw the inbound potential,in_i, for port i between 0.2 and 1.8 uniformly, while the corresponding out_j is set as $out_j = 2 - in_i$. The reason for this setting is that in real-world regions, large inbound flows port usually exhibits small outbound flows. We also include a random number v with mean 30, that is, the typical daily demand between each pair of locations to capture the randomness in demand. In order to test the performance of the algorithms under different distributions, we also evaluate the performance under exponential, normal and uniform distribution. We set the stepsize α_k to $1/k$.

We solve a deterministic network flow problem to construct an initial piecewise linear functions as described in [1], and we replace the random demand by their mean values in the deterministic problem. Then, we can obtain $\overline{S} = \{\overline{S}_1, \overline{S}_2, \dots, \overline{S}_n\}$. For each $i \in L$, we generate the initial approximation function $\widehat{Q}_i^0(x) = c(x - \overline{S}_i)^2, x = 0, \delta, \dots, k\delta, \dots .K\delta$, where c is a positive parameter and $x \in [0, K\delta]$. In the projection step, a least-squares problem is solved as following:

$$x^{k+1} = \operatorname{argmin}\left(x^{k+1} - \left(x^k + \alpha_k g^k\left(x^k\right)\right)\right)^2, x^{k+1} \in X.$$

4.2 Effectiveness and efficiency performance

To test the efficiency of the algorithm, we use a myopic algorithm, a posterior bound (PB), the L-shaped algorithm [12] and the inexact cut algorithm [15] as benchmarks. The myopic algorithm simply solves a static deterministic assignment problem at the current stage while ignoring uncertainties in the second stage. It is necessary to solve a deterministic network flow problem with all realized demands to obtain PB. Note that such a posterior optimization involves no uncertainty since decisions are allowed to anticipate future demand. Thus, the cost of PB is the lowest and normally unreachable. As for the L-shaped algorithm and the inexact cut algorithm, a group of linear programming problems with valid cuts should be solved.

We use 8 instances, in which the number of empty containers is ranged from 400 to 3200, and the corresponding number of ports is ranged from 5 to 40. For each instance, 2000 samples are implemented and we obtain the solutions of the myopic algorithm and the sample means of PB, the inexact cut algorithm, the L-shaped algorithm, SHLA and ProSHLA. For SHLA, two classes of initial functions with δ = 1 and δ = 2 are selected, whereas we select δ = 2 for ProSHLA.

We show the experiment results on total cost in **Table 1**. In **Table 1**, column 1 presents the number of ports, and column 2 shows the number of the empty containers. The PB bounds are contained in column 3. Columns 4–9 contain the solutions achieved by the myopic algorithm, the L-shaped algorithm, the inexact cut algorithm, SHLA-1, SHLA-2 and ProSHLA, respectively. From the table, it clearly demonstrates that the inexact cut algorithm, the L-shaped algorithm, SHLA, and ProSHLA can achieve optimal or very-near-optimal solutions, which are closer to the PB (lowest) bounds than those of the myopic method. Moreover, the solutions of the L-shaped algorithm are the best solutions known, which are slightly better than the inexact cut

N_L	N_R	Total cost (dollars)						
		PB	**Myopic**	**L-shaped**	**Inexact cut**	**SHLA-1**	**SHLA-2**	**ProSHLA**
5	400	−28,551,302	−27,761,331	−28,551,274	−28,551,227	−28,464,248	−28,463,941	−28,464,002
10	800	−59,397,423	−58,432,159	−59,396,702	−59,396,319	−59,338,653	−59,338,190	−59,338,341
15	1200	−98,451,193	−93,188,576	−98,449,868	−98,449,427	−98,257,484	−98,257,244	−98,257,395
20	1600	−147,390,005	−141,062,187	−147,388,360	−147,387,532	−147,269,239	−147,269,212	−147,269,213
25	2000	−185,223,883	−180,875,614	−185,220,423	−185,219,663	−185,092,289	−185,092,113	−185,092,143
30	2400	−234,005,740	−226,978,090	−234,004,427	−234,003,513	−233,401,849	−233,401,516	−233,401,658
35	2800	−266,375,728	−260,966,356	−266,375,468	−266,374,497	−266,337,862	−266,337,649	−266,337,660
40	3200	−304,910,355	−293,882,881	−304,908,597	−304,906,829	−304,907,153	−304,906,829	−304,906,962

Table 1.
Total cost for SHLA and ProSHLA.

N_R	N_L	**Computation time (s)**				
		Inexact cut	**L-shaped**	**ProSHLA**	**SHLA-1**	**SHLA-2**
400	5	76	153	43	28	28
800	10	382	535	148	90	95
1200	15	598	1140	332	224	217
1600	20	1084	2277	628	417	420
2000	25	1843	4190	1107	676	790
2400	30	2338	5491	1559	1112	1066
2800	35	4249	8075	2341	1539	1531
3200	40	8154	18,636	5307	3010	3428

Table 2.
Computational time of ProSHLA and SHLA.

algorithm because the latter produces valid cuts that are inexact in the sense that they are not as constraining as optimality cuts in the Lshaped algorithm. In addition, the performance of SHLA (δ = 1) outperforms that of SHLA (δ = 2) and ProSHLA (δ = 2), the reason is that small δ can lead to good performance. A specific discussion with impact ofδ will be demonstrated later. The performance of ProSHLA (δ = 2) is slight better than SHLA (δ = 2). Because the projection steps in ProSHLA help improve the solution. Considering the speed of convergence is quite important in practical problems, we will focus on the computational time for different algorithms, which is shown in the **Table 2** below.

As shown in **Table 2**, ProSHLA and SHLA are more efficient than the inexact cut algorithm and the L-shaped algorithm because ProSHLA and SHLA can utilize the network structure while using the stochastic sub-gradient to approximate the recourse function. From the table, we find that the inexact cut algorithm and L-shaped algorithm are time-consuming, the reason is that here are 2000 samples, and it corresponds to a very large number of cuts for the inexact cut algorithm and L-shaped algorithm. It can also be observed that the computational time of the inexact cut algorithm is smaller than that of the L-shaped algorithm, and the reason is that the optimality cut in L-shaped is more than the valid cuts in the inexact cut algorithm. Moreover, the computational time of SHLA (δ = 1) is almost equal to that of (δ = 2), which reveals that the computational time canot be affect by the choice of δ. In contrary, ProSHLA(δ = 2) requires more computational time than SHLA (δ = 2), and the reason is that the projection step in ProSHLA are time-consuming. In the following text, we focus on the convergence performance of SHLA and ProSHLA. Thus, only the results of the myopic algorithm, PB, ProSHLA and SHLA are demonstrated, and we use the solutions of the myopic algorithm and PB as the upper and lower bounds, respectively.

4.3 Analysis of convergence performance

In this subsection, a set of experiments are conducted to evaluate the convergence performance of SHLA and ProSHLA, and we choose the second instance (N_R = 800 and N_L = 10) as the experimental illustration. The range of the sample number is set

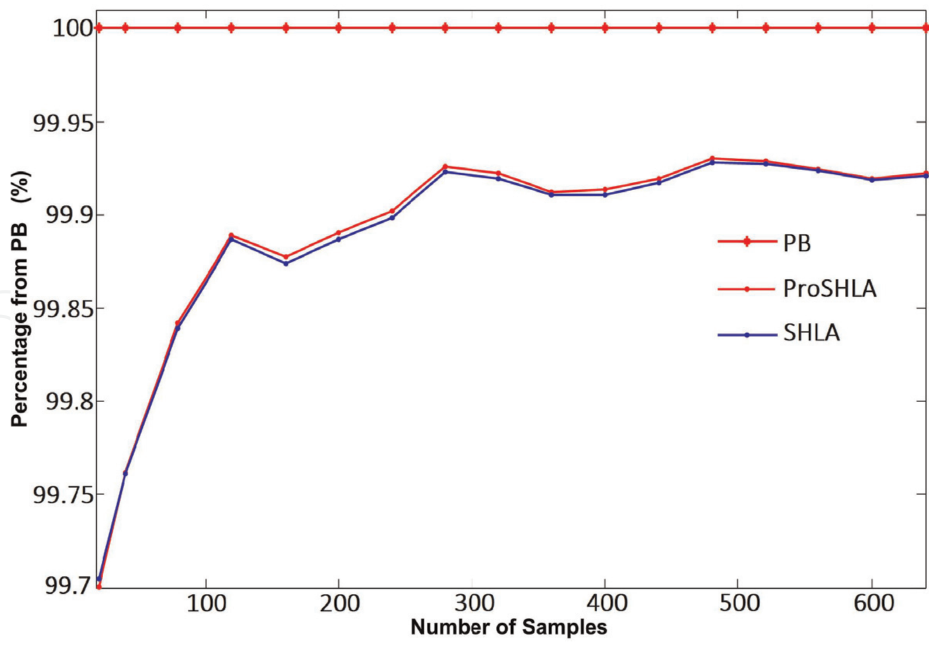


Figure 3.
Convergence rate of ProSHLA and SHLA.

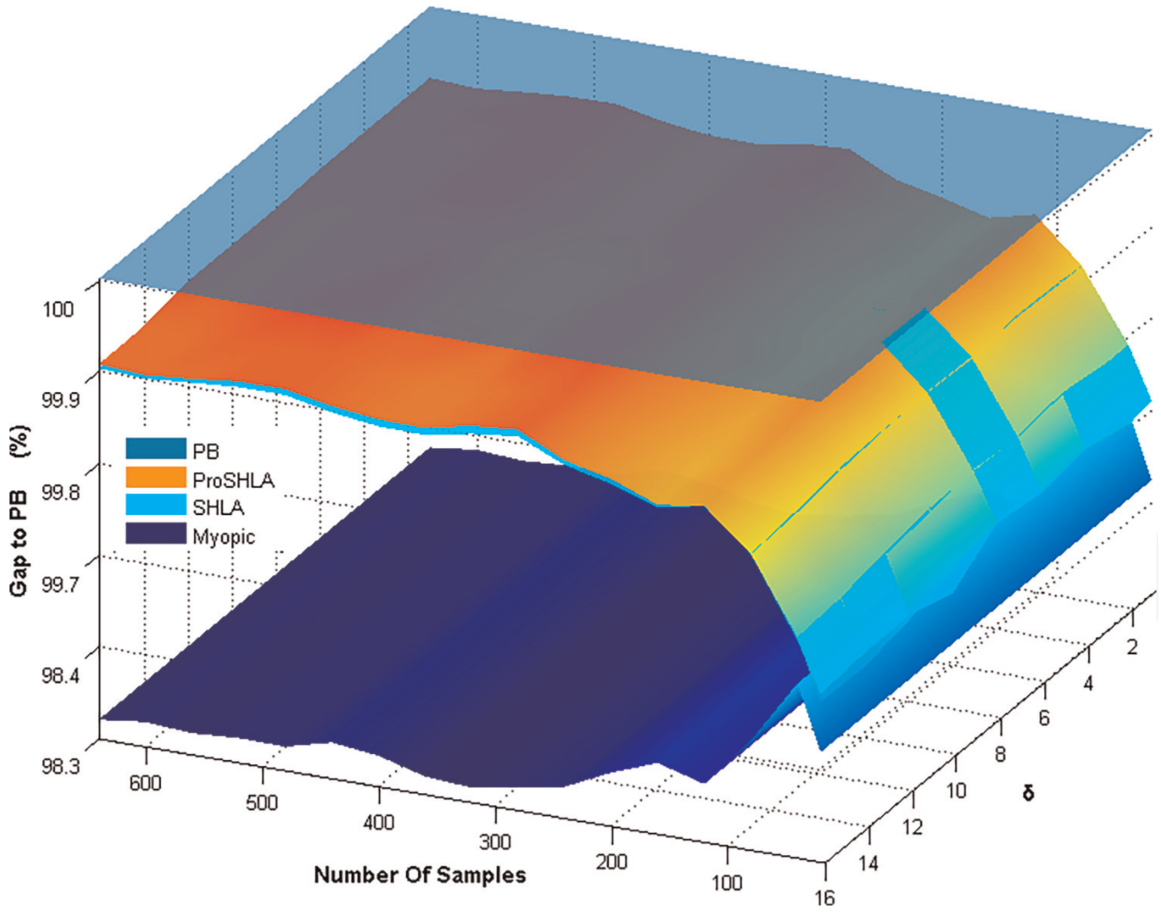


Figure 4.
Gaps to PB for various δ.

from 20 to 640, and we record the result of each combination of N_R and N_L at each iteration. We can seem from **Figure 3** that the convergence rate of SHLA-1 and ProSHLA is remarkably high.

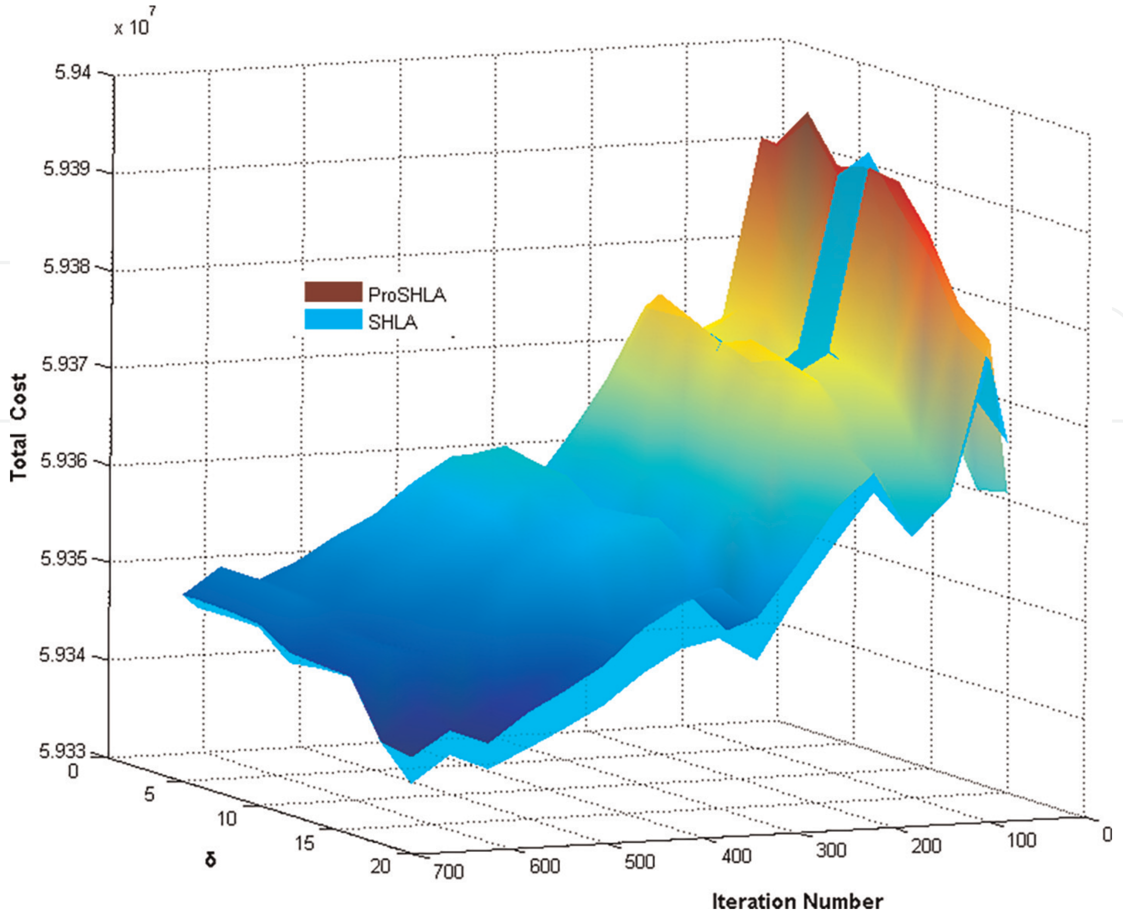


Figure 5.
Comparison of ProSHLA and SHLA for various δ.

To further evaluate how δ affects the algorithm's convergence performance, a set of computational experiments are conducted. We increase δ from 1 to 16 and the number of samples from 20 to 640. Here are many combination of δ and the number of samples. We record the sample means of the solutions of SHLA and ProSHLA, PB and the myopic method for each combination. We demonstrate the 3D plots of the solution in **Figures 4** and **5**. As in **Figure 4**, the layers of ProSHLA and SHLA are extremely close to the PB layer, and this implies the ProSHLA and SHLA are convergent rapidly for various δ. Furthermore, it can been seem that the performance of ProSHLA can slightly exceeds that of SHLA. In order to further investigate the difference between SHLA and ProSHLA, we demonstrate the performance of ProSHLA and SHLA separately in **Figure 5** (without the myopic algorithm and PB). As described in **Figure 5**, the choice of δ can affect the performance of ProSHLA and SHLA, and a small δ usually leads to a good solution.

We provides more details on the convergence performance of ProSHLA and SHLA for various δ in **Table 3** below, which clearly demonstrates that in conjunction with the small δ, the performance of SHLA and ProSHLA is close to that of PB.

4.4 An extension experiment on a high dimensional resource allocation problem

Due to the limitation of the container setting, an extension experiment on a higher dimensional problem is considered in this subsection. In this problem, there exists several retailers R and many production facilities (with warehouse) L. In stage 1, an

Total cost—% gap to PB (computational time in seconds)				
δ	PB	Myopic	SHLA	ProSHLA
1	−59,397,423	1.6251%	0.0989%(909.5)	0.0989% (1441.5)
2	−59,397,423	1.6251%	0.0997%(907.9)	0.0995% (1506.5)
4	−59,397,423	1.6251%	0.1001%(901.8)	0.0997% (1503.6)
6	−59,397,423	1.6251%	0.1005%(911.5)	0.1004% (1553.5)
8	−59,397,423	1.6251%	0.1028%(901.7)	0.1024% (1535.6)
16	−59,397,423	1.6251%	0.1189%(920.3)	0.1150% (1565.4)

Table 3.
Performance under various δ (no. of samples is 2000).

amount x_{ij} is moved to a warehouse or retailer or location j from production facility i before the retail demand is realized. When we know the consumer's demand, then y_{ij} products are moved to retailer location j from production facility i. Besides, the type of the consumer's demand at each location j is different, we denote the type as $t \in$ ☐✕, we set the consumer's demand of type t at location j as D_j^t, and provide p_i^t unit of type t at production location i. We denote the production capacity of location i by cap_i. This problem is a non-separable problem.

Subsequently, we formulate the problem as follows:

$$min \sum_{i\in L}\sum_{j\in L\cup R} c_{ij}^1 y_{ij} + E_\omega[Q(x,\ \omega)] \tag{38}$$

subject to

$$\sum_{j\in L\cup R} x_{ij} \le cap_i, \qquad \forall i\in L \tag{39}$$

$$\sum_{i\in L} x_{ij} = s_j, \qquad \forall j\in L\cup R \tag{40}$$

$$x_{ij}, s_j \ge 0,\ \ \forall i\in L, \forall j\in L\cup R \tag{41}$$

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

$$Q(x, \omega) = min \sum_{i\in L\cup R}\sum_{j\in R} c_{ij}^2 y_{ij} - \sum_{i\in R}\sum_{t\in T} r_i^t p_i^t \tag{42}$$

subject to

$$\sum_{j\in R} y_{ij} = s_i, \qquad \forall i\in L\cup R \tag{43}$$

$$\sum_{i\in L\cup R} y_{ij} = \sum_{t\in T} p_j^t,\ \ \forall i\in R \tag{44}$$

$$p_j^t \le D_j^t(\omega),\ \ \forall t\in L\cup R, \forall j\in R, t\in T \tag{45}$$

In the first stage, we set$c_{ij}^1 = c_0^1 + c_1^1 d_{ij}$, where d_{ij} is the Euclidean distance between locations i and j, andc_0^1 is the production cost for each product and c_1^1 is the transportation cost per mile. For the second stage costs, we set

$$c_{ij}^2 = \begin{cases} c_1^2 d_{ij} & \text{if } i \in L \text{ or } i = j \\ c_0^2 + c_1^2 d_{ij} & \text{if } i \in R \text{ and } i \neq j \end{cases}$$

c_1^2 is the transportation cost per mile in the second stage, and c_0^2represents the fixed charge for moving each product from one retailer location to another retailer location. For one unit of the demand type t occurring in retailer location i, a revenue r_i^t will be obtained. Our problem instances differ in the number of products and $|L \cup R|$, and it determines the dimensionality of the recourse function.

Similarly, we use the inexact cut algorithm [15] and the L-shaped algorithm [12] as benchmarks, and these two algorithms are Benders decomposition based methods. Considering the convergence rate is quite important practically, in this part, our main focus is on the speed of convergence. In order to evaluate the speed of convergence of different methods, each algorithm is implemented for 40, 160, 640, 1200, and 4000 iterations, and a side by side comparison of the algorithms has been made when the number of iteration increases. For the L-shaped and inexact cut algorithms, the number of iterations refer to the number of cuts used to approximate the expected recourse function. For ProSHLA (δ = 2), the number of iterations refer to the number of demand samples used.

Table 4 below shows the experiment results. In the experiment, the L-shaped algorithm has been used to help find the optimal solution. In the table, the numbers denote the percent deviation between the optimal value and the objective value.

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 objective value and the optimal value obtained after a certain number of iterations. The computational time per iteration are also listed in **Table 4**. The computational results on 5 scale of dimensionality instances.

In **Table 4** above, column 1 presents the number of the locations, and column 2 shows the number of the products. Column 3 presnets the method that we used in the experiment. The percent deviation from the optimal value are contained in columns 4 to 8. Column 9 lists the computational time per iteration. According to results in **Table 4**, ProSHLA is able to obtain high quality solutions very efficient for different problem instance, and it can maintain the consistent performance in problem of different sizes, especially for large problems. This performance characteristic makes ProSHLA promising for large-scale application. In comparison with these two Benders decomposition-based methods, ProSHLA is competitive for high dimensional problems. The reason is that separable approximations usually scale much more easily to very high dimensional problems. Note that in the first problem instance, when the number of location is 6 and the number of resource is 10 (the inventory in a location might be 0, 1, 2), the result of ProSHLA seems to be breakdown, because the problem instance in this subsection is non-separable, which may introduce errors when we use the separable approximations to approximate the expected recourse function. However, it will not happen on large problems. As for large problems, the separable approximations are nearly continuous, rather than being just piecewise continuous.

According to the above computational results, ProSHLA is a promising method for two-stage stochastic programs, but more comprehensive numerical work is needed

			Number of iterations					
$N_{\vert L \cup R \vert}$	N_P	***Algorithm***	**40**	**160**	**640**	**1200**	**4000**	**Sec./iter.**
|L ∪ R| = 6	10	ProSHLA	13.26	8.61	2.93	2.92	2.73	0.01
		L-shaped	1.17	0	0	0	0	0.07
		Inexact cut	0.96	0	0	0	0	0.05
|L ∪ R| = 10	200	ProSHLA	10.58	3.01	0.61	0.29	0.12	0.07
		L-shaped	1.85	0	0	0	0	0.26
		Inexact cut	1.31	0	0	0	0	0.21
|L ∪ R| = 20	400	ProSHLA	7.22	1.22	0.42	0.23	0.05	0.30
		L-shaped	10.46	1.16	0	0	0	1.13
		Inexact cut	6.63	0.98	0	0	0	1.04
|L ∪ R| = 40	800	ProSHLA	6.03	0.82	0.34	0.17	0.02	0.83
		L-shaped	23.57	3.23	0.31	0.02	0	9.53
		Inexact cut	17.16	2.24	0.13	0.01	0	8.72
|L ∪ R| = 100*	2000	ProSHLA	5.68	0.78	0.15	0.04	0 0.03	2.68
		L-shaped	44.84	14.51	1.38	0.5		36.53
		Inexact cut	29.56	8.14	0.91	0.25	0.03	30.98

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

Table 4.
Percent error over optimal solution with different algorithms costs.

before using it in a particular problem. Owing to its efficient performance and simplicity, ProSHLA is a very promising candidate for high-dimensional problems. Moreover, we can use it as an initialization routine method for high-dimensional stochastic programming problems, and it can exploit high-quality initial feasible solution.

5. Conclusion

In this study, we propose an efficient machine learning algorithm for two-stage stochastic programs. This machine learning algorithm is termed as projected stochastic hybrid learning algorithm, and consists of stochastic sub-gradient and piecewise linear approximation methods. We use the stochastic sub-gradient and sample information to update the piecewise linear approximation on the objective function. Then we introduce a projection step, which implemented the sub-gradient methods, to jump out from a local optimum, so that we can achieve a global optimum. By the innovative projection step, we show the convergent property of the algorithm for general two-stage stochastic programs. Furthermore, for the network recourse problem, our algorithm can drop the projection steps, but still maintains the convergence property. The computational results reveal the efficiency of the proposed algorithms and the proposed algorithms are distribution-free. Furthermore, the convergence rate can be affected by the granularity of the initial function (δ). Small granularity usually leads to a high convergence rate. Finally, the computational results also show that the proposed algorithm is very competitive for high dimensional problems. Compared with MAT, the proposed algorithm can collect the information based on knowledge gradients and use it to update the recourse function by learning steps. It can overcome the "curse of dimensionality". Moreover, it can transfer the problem into a polynomial solvable problem.

Acknowledgements

This research is partially supported by the National Nature Science Foundation of China (Project No. 71701221) and the Natural Science Foundation of Guangdong Province, China, under Grant number: 2019A1515011127.

Author details

Zhou Shaorui*, Cai Ming and Zhuo Xiaopo
Sun Yat-sen University, Guangzhou, China

*Address all correspondence to: zshaorui@gmail.com

References

[1] Cheung RK, Chen CY. A two-stage stochastic network model and solution methods for the dynamic empty container allocation problem. Transportation Science. 1998;**32**(2): 142-162

[2] Bouzaiene-Ayari B, Cheng C, Das S, Fiorillo R, Powell WB. From single commodity to multiattribute models for locomotive optimization: A comparison of optimal integer programming and approximate dynamic programming. Transportation Science. 2016;**50**:366-389

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

[4] Kim K, Mehrotra S. A two-stage stochastic integer programming approach to integrated staffing and scheduling with application to nurse management. Operations Research. 2015;**63**:1431-1451

[5] Wallace SW, Ziemba WT. Applications of stochastic programming. In: MOS-SIAM Series on Optimization. Vol. 5. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM); Mathematical Programming Society (MPS); 2005. ISBN:0-8971-555-5

[6] Kleywegt AJ, Shapiro A. Homem de Mello T. the sample average approximation method for stochastic discrete optimization. SIAM Journal on Optimization. 2001;**12**(2):479-502

[7] Ermoliev Y. Stochastic quasigradient methods. In: Numerical Techniques for Stochastic Optimization. New York: Springer-Verlag; 1988

[8] Robbins H, Monro S. A stochastic approximation method. The Annals of Mathematical Statistics. 1951;**22**(3): 400-407

[9] Rockafellar RT, Wets JB. A note about projections in the implementation of stochastic quasigradient methods. In: Numerical Techniques for Stochastic Optimization, Springer Ser. Comput. Math. Vol. 10. Berlin: Springer; 1988. pp. 385-392

[10] Ruszczyñski A. A linearization method for nonsmooth stochastic optimization problems. Mathematics of Operations Research. 1987;**12**:32-49

[11] Benders JF. Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik. 1962;**4**(1):238-252

[12] Van Slyke RM, Wets RJ-B. L-shaped linear programs with applications to optimal control and stochastic programming. SIAM Journal on Applied Mathematics. 1969;**17**(4):638-663

[13] Pereira MVF, Pinto LMVG. Multi-stage stochastic optimization applied to energy planning. Mathematical Programming. 1991;**52**:359-375

[14] Zakeri G, Philpott AB, Ryan DM. Inexact cuts in benders decomposition. SIAM Journal on Optimization. 2000; **10**(4):643-657

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

[16] Rebennack S. Combining sampling-based and scenario-based nested benders decomposition methods: Application to stochastic dual dynamic programming.

Mathematical Programming. 2016; **156**(1):343-389

[17] Philpott AB, Guan Z. On the convergence of stochastic dual dynamic programming and related methods. Operations Research Letters. 2008;**36**: 450-455

[18] Girardeau P, Leclere V, Philpott AB. On the convergence of decomposition methods for multistage stochastic convex programs. Mathematics of Operations Research. 2015;**40**(1): 130-145

[19] Cheung RK, Powell WB. SHAPE—A stochastic hybrid approximation procedure for two-stage stochastic programs. Operations Research. 2000; **48**(1):73-79

[20] Powell WB, Ruszczyñski A, Togaloglu H. Learning algorithms for separable approximation of discrete stochastic optimization problems. Mathematics of Operations Research. 2004;**29**(4):814-836

[21] Godfrey GA, Powell WB. An adaptive dynamic programming algorithm for dynamic fleet management I: Single period travel times. Transportation Science. 2002;**36**(1):21-39

[22] Neveu J. Discrete Parameter Martingales. Amsterdam: North Holland; 1975

[23] Song DP, Dong JX. Empty container management in cyclic shipping routes. Maritime Economics & Logistics. 2008; **10**(4):335-361

[24] Zhou S, Zhang H, Shi N, Xu Z, Wang F. A new convergent hybrid learning algorithm for two-stage stochastic programs. European Journal of Operational Research. 2020;**283**(1): 33-46

[25] Xu L, Zou Z, Zhou S. The influence of COVID-19 epidemic on BDI volatility: An evidence from GARCH-MIDAS model. Ocean Coastal Management. 2022. DOI: 10.1016/j.ocecoaman. 2022.106330