Benders Cut Classification via Support Vector Machines for Solving Two-stage Stochastic Programs

Huiwen Jia
Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, hwjia@umich.edu;
Siqian Shen
Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, siqian@umich.edu.

We consider Benders decomposition for solving two-stage stochastic programs with complete recourse and finite samples of uncertain parameters. We define the Benders cuts binding at the final optimal solution or the ones significantly improving bounds over iterations as valuable cuts. We propose a learning-enhanced Benders decomposition (LearnBD) algorithm, which adds a cut classification step in each iteration to learn cuts that are more likely to be valuable cuts. The LearnBD algorithm includes three phases: (i) cut sampling, (ii) cut classifier construction using support vector machines (SVM), and (iii) cut classification. We run the LearnBD algorithm on instances of capacitated facility location problems under uncertain demand. Our results show that SVM cut classifier works effectively for identifying valuable cuts and the LearnBD algorithm improves the overall computational efficiency for randomly generated instances with various sizes and complexities.

Key words: Benders decomposition, two-stage stochastic (integer) programming, support vector machine (SVM), cut classification

1. Introduction

In this paper, we focus on the Benders decomposition algorithm (Benders [1962]) and its implementation for solving a broad class of two-stage stochastic programming models with a finite number of realizations of uncertain parameters, called “scenarios” or “samples” in the stochastic programming literature (see, e.g., Birge and Louveaux [2011]). The decision-making processes are separated into two stages. In the first stage, decision variables $x \in \mathbb{R}^{n_1}$, or $x \in \mathbb{Z}^{n_1}$, are made before realizing the uncertainty. In the second stage, decision variables $y \in \mathbb{R}^{n_2}$ are recourse decisions, dependent on the values of $x$-variables and uncertain parameter realized in each scenario. In our problems, the first-stage variables $x$ can be integer or continuous, and their solution values are restricted by only linear constraints. The second-stage problems are linear programs associated with individual scenarios. The second-stage variables $y$ are linked with the first-stage variables $x$ via linear constraints defined as $Wy = h - Tx$ and the cost function in the second-stage problem is $q^Ty$, where matrix $W \in \mathbb{R}^{m_2 \times n_2}$, vector $h \in \mathbb{R}^{m_2}$, matrix $T \in \mathbb{R}^{m_2 \times n_1}$ and cost vector $q \in \mathbb{R}^{n_2}$ can be subject to uncertainty.
Let \( \Omega \) be the sample space that contains all the scenarios, where scenario \( \omega \) is associated with a specific realization of uncertain parameter \( (W_\omega, h_\omega, T_\omega, q_\omega) \) and has an occurrence probability \( p_\omega \) such that \( \sum_{\omega \in \Omega} p_\omega = 1 \). The cost of future recourse solution in scenario \( \omega \), \( \forall \omega \in \Omega \), with fixed first-stage variables \( x \), is given by:

\[
Q_\omega(x) \overset{\text{def}}{=} \min_y q_\omega^T y \\
\text{s.t.} \quad W_\omega y = h_\omega - T_\omega x.
\] (1)

The goal is to minimize a linear cost function of the first-stage solution, \( c^T x \), plus an expected cost of future recourse solution as \( \mathbb{E}_{\mathbb{P}}(Q_\omega(x)) = \sum_{\omega \in \Omega} p_\omega Q_\omega(x) \). We denote the feasible region of \( x \) in the first stage as \( X \), and model the overall problem as

\[
\min_{x \in X} c^T x + \sum_{\omega \in \Omega} p_\omega Q_\omega(x).
\] (2)

Note that the assumption of having finite scenarios is made without loss of generality, because if the uncertain parameter \( (W, h, T, q) \) follows a continuous distribution \( \mathbb{P} \), we can apply the Monte Carlo sampling approach to generate \( N_s \) i.i.d. samples \( \{\omega_1, \ldots, \omega_{N_s}\} \) of uncertain parameter. Then we apply the sample average approximation (SAA) \( \frac{1}{N_s} \sum_{i=1}^{N_s} Q_{\omega_i}(x) \) to replace \( \mathbb{E}_{\mathbb{P}}(Q_\omega(x)) \). Kleywegt et al. (2002) prove that both the optimal solution and the optimal objective value of the SAA model will converge to those of the original problem, when \( N_s \) grows sufficiently large.

1.1. Generic Benders Decomposition

Given a large number of scenarios, problem \( [2] \) is a large-scale integer program and can be computationally intractable. The Benders decomposition algorithm, which takes the advantage of the decomposable structure of two-stage stochastic programs, is applied widely to optimize variants of problem \([2]\) formulated for a wide variety of applications (see, e.g., Magnanti and Wong 1981). Creating new variables \( \theta_\omega \in \mathbb{R}, \omega \in \Omega \) in the first-stage problem, one can formulate a relaxation of the original problem, called relaxed master problem (RMP), which has an initial form:

\[
(RMP^0) \quad \min_{x \in X, \theta} c^T x + \sum_{\omega \in \Omega} p_\omega \theta_\omega.
\] (3)

Subproblems (SPs) are defined as the linear programming dual of the second-stage problems \([1]\) with dual variables \( \pi_\omega \in \mathbb{R}^{m_2}, \forall \omega \in \Omega \). We refer to SP\( _\omega \) as the SP for scenario \( \omega \), formulated as

\[
(SP_\omega) \quad Q^D_\omega(x) = \max_{\pi_\omega} (h_\omega - T_\omega x)^T \pi_\omega \\
\text{s.t.} \quad W_\omega^T \pi_\omega \leq q_\omega.
\] (4)

Let \( V^{\omega,t} \) be the set of identified extreme points of the feasible region of \( SP_\omega \) in iteration \( t \), and we have \( V^{\omega,0} = \emptyset \). Similarly, let \( R^{\omega,t} \) be the set of identified extreme rays of the feasible region of \( SP_\omega \) in iteration \( t \), and \( R^{\omega,0} = \emptyset \). The two sets are respectively associated with Benders optimality
cuts and feasibility cuts generated during iterations 1, \ldots, t - 1, and we will explain the cuts details later. In iteration $t$, the corresponding RMP is given by:

\[
(RMP^t) \quad \min_{x \in X, \theta} \quad c^T x + \sum_{\omega \in \Omega} p_\omega \theta_\omega \\
\text{s.t.} \quad \theta_\omega \geq (h_\omega - T_\omega x)^T \nu_\omega \quad \omega \in \Omega, \quad \nu_\omega \in V^{\omega,t}; \\
(h_\omega - T_\omega x)^T \rho_\omega \leq 0 \quad \omega \in \Omega, \quad \rho_\omega \in R^{\omega,t}. \tag{5}
\]

We solve $RMP^t$ and obtain an optimal solution $(\hat{x}_t, \hat{\theta}_t)$. Then for each scenario $\omega \in \Omega$ and its subproblem $SP_\omega$, we first check whether the current solution leads to a feasible second-stage problem by solving a corresponding problem $SP_\omega$-F with decision variables $\sigma_\omega \in \mathbb{R}^{m_2}$, modeled as

\[
(SP_\omega\text{-}F) \quad \max_{\sigma_\omega} \quad (h_\omega - T_\omega \hat{x}_t)^T \sigma_\omega \\
\text{s.t.} \quad W_\omega^T \sigma_\omega \leq 0; \\
\|\sigma_\omega\| \leq 1. \tag{6}
\]

If $SP_\omega$-F has a positive optimal objective value with an optimal solution $\bar{\sigma}_\omega$, then from any feasible solution to $SP_\omega$, we can move along the direction $\bar{\sigma}_\omega$ to stay feasible (due to the first constraint of model (6)) but increase the objective value of $SP_\omega$. This implies that $SP_\omega$ is unbounded and the second-stage problem is infeasible for given $\hat{x}_t$. To cut off the infeasible first-stage solution $\hat{x}_t$, we generate a Benders feasibility cut:

\[
(h_\omega - T_\omega x)^T \bar{\sigma}_\omega \leq 0 \tag{7}
\]

to $RMP^{t+1}$, which is equivalent to letting $R^{\omega,t+1} = R^{\omega,t} \cup \{\bar{\sigma}_\omega\}$. In this paper, we only focus on the case having complete recourse, under which any feasible first-stage solution will result in a feasible second-stage problem in each scenario. Therefore, our $RMP^t$ (i.e., Model (5)) for each iteration $t$ only contains the first set of optimality cuts, whose derivation is given as follows. If the subproblem is feasible, then in iteration $t$ we check the optimality of $(\hat{x}_t, \hat{\theta}_t)$. We solve $SP_\omega$ with $x = \hat{x}_t$ to obtain an optimal solution $\bar{\pi}_\omega$ and the optimal objective value $Q^D_\omega(x) = \bar{\pi}_\omega^T (h_\omega - T_\omega \hat{x}_t)$. By strong duality we know for any value of $x$, $Q^D_\omega(x) = Q_\omega(x)$. The solution to $RMP^t$ will reach the same objective value as the original problem when $\hat{\theta}_\omega \geq Q_\omega(\hat{x}_t), \quad \omega \in \Omega$. Therefore, $\hat{\theta}_\omega < Q^D_\omega(x)$ indicates that the current solution $(\hat{x}_t, \hat{\theta}_t)$ is not optimal to the original problem. Thus, we add a Benders optimality cut

\[
\theta_\omega \geq (\bar{\pi}_\omega)^T (h_\omega - T_\omega x) \tag{8}
\]

to $RMP^{t+1}$, which is equivalent to letting $V^{\omega,t+1} = V^{\omega,t} \cup \{\bar{\pi}_\omega\}$.

In iteration $t$, the objective value of $RMP^t$ provides a valid lower bound to the origin problem because it is a relaxation. If all SPs have finite optimal objective values, $\hat{x}_t$ and all recourse solutions together form a feasible solution to the original two-stage problem, and thus

\[
c^T \hat{x}_t + \sum_{\omega \in \Omega} p_\omega Q_\omega(\hat{x}_t)
\]
provides a valid upper bound. The algorithm terminates when the upper and lower bounds are equal or their gap is within a pre-specified tolerance $\delta$. Here, we define the gap as

$$\text{optimality gap} = \frac{\text{upper bound} - \text{lower bound}}{\text{lower bound}} \times 100\%.$$  \hspace{1cm} (9)

The Benders decomposition algorithm converges in a finite number of iterations due to the finite number of dual extreme rays and extreme points of the finitely many SPs.

1.2. Challenges and Research Overview

The Benders decomposition method helps to solve two-stage stochastic programs efficiently, however, it could suffer from slow convergence. One reason is that the size of RMPs becomes too large due to the quickly increased number of newly added cuts over iterations. Geoffrion and Graves (1974) are among the first to notice and emphasize on the computational difficulty of solving RMPs for stochastic binary integer programs. Magnanti and Wong (1981) report that over 90% of the total time of implementing the Benders decomposition is spent on solving RMPs. Minoux (1986) points out that not all extreme points of the feasible region of SPs equally contribute to restricting the optimal solution to RMPs. Therefore, a larger number of Benders cuts are not tight at the final optimal solution, but can increase the size of RMPs.

In this paper, we propose a three-phase learning-enhanced Benders decomposition (LearnBD) algorithm to solve two-stage stochastic integer programs with finite samples of the uncertain parameter and complete recourse, where the second-stage subproblems are assumed to be convex. We define cuts as valuable cuts when they can either cut the feasible region in the current iteration significantly, or be tight at the final optimal solution (see Holmberg (1990) for a similar definition in the latter case). Our goal is to only add valuable cuts to the corresponding RMP in each iteration. Up to date, there is no practical and systematic way to perform cut classification and to accelerate the iteration process for Benders decomposition for large-scale optimization problems, according to Rahmanian et al. (2017). We propose to integrate machine learning techniques into the traditional Benders decomposition framework to learn cut characteristics and selectively generate subsets of Benders cuts iteratively.

1.3. Contribution of the Paper

We summarize the main contributions of this paper as follows.

1. We identify a set of characteristics and quantify performance measures of Benders cuts. We construct a cut classifier using support vector machines (SVM), a widely used supervised machine learning method that takes history observations and their labels as input, to identify valuable cuts in each iteration.
2. We develop the LearnBD algorithm with SVM cut classifier, to limit the size of RMPs and reduce total solving time. We also provide guideline for choosing hyperparameters for enhancing the effectiveness of the LearnBD algorithm.

3. We test instances of capacitated facility location problem with random demand, to demonstrate the computational results in different problem settings. Our results show that the LearnBD algorithm leads to smaller sizes of RMPs with fewer accumulated cuts, shorter cumulative time for solving RMPs, and a higher percentage of tight cuts in the last iteration.

1.4. Structure of the Paper
The remainder of this paper is organized as follows. In Section 2, we review the literature on the effort of improving Benders decomposition for solving different types of problems. In Section 3, we develop the LearnBD algorithm and use SVM for constructing the cut classifier. In Section 4, we present the computational results of the LearnBD algorithm benchmarked with the traditional Benders approach. In Section 5, we conclude the paper and describe future research directions.

2. Literature Review
The Benders decomposition algorithm was initially proposed by Benders (1962) and was then widely used for solving problems of scheduling and planning (Cordeau et al., 2001; Hooker, 2007), network flow optimization and transportation (Laporte et al., 1994; Costa, 2005; Binato et al., 2001), and inventory control and resource management (Federgruen and Zipkin, 1984; Cai et al., 2001). Meanwhile, some general-purpose work, such as Magnanti and Wong (1981) and Naoum-Sawaya and Elhedhli (2013), conclude that directly applying the traditional Benders decomposition may require excessive computational resources. It is mainly due to the poor convergence of RMPs that has been computationally demonstrated in Orchard-Hays et al. (1968) and Wolfe (1970). Various researchers have proposed enhancement strategies for different types of problems to accelerate the algorithm accordingly, of which we describe the details as follows.

In the traditional Benders approach detailed in Section 1.1, we solve RMPs and SPs iteratively, and thus the first stream of studies concentrates on problem-solving techniques, and in particular techniques for efficiently computing RMPs or SPs. Geoffrion and Graves (1974) propose to only sub-optimally solve RMPs in each iteration to enable cut generation, without seeking tight cuts at the beginning of the Benders approach. Similarly, Raidl (2015) solves RMPs using heuristics to save computational time. Zakeri et al. (2000) show that sub-optimal solutions to the SPs can still generate valid cuts in RMPs, and thus effective heuristic approaches are designed for solving the SPs approximately.

The second stream of studies focuses on decomposition strategies, to guide the process of partitioning variables to remain in RMPs or in SPs. Crainic et al. (2014) point out that RMPs, which
only contain part of the decision variables, lose important information about the problem and accordingly may lead to many iterations before converging. Therefore, they propose a so-called Partial Benders Decomposition to reduce the number of feasibility and optimality cuts, while adding information of SPs into RMPs by retaining or creating scenarios. They develop different decomposition strategies for choosing the retaining scenario for solving two-stage mixed-integer programming (MIP) models with continuous recourse. They prove that the proposed approach can reduce the need for generating feasibility cuts. In addition, Gendron et al. (2016) propose a non-standard decomposition strategy, which retains the second-stage variables in RMPs, and the authors test the results using instances of network design problems.

Machine learning techniques have been applied to general-purpose optimization algorithms for urging quick convergence to optimal or sub-optimal solutions (see, e.g., He et al. 2014; Khalil et al. 2017; Khalil et al. 2016) introduce a novel data-driven framework for variable selection to efficiently solve MIP models via branch-and-bound algorithm. Kruber et al. (2017) develop a supervised learning approach to distinguish a stronger reformulation of a given MIP model and to determine which decomposition to implement in order to improve the speed of MIP solvers. To the best of our knowledge, machine learning techniques have not been integrated into standard decomposition algorithms including Benders, which is the main motive for this work.

3. Learning-enhanced Benders Decomposition Algorithm

The overview of the LearnBD algorithm is to first construct a cut classifier for a given problem and then apply cut classification steps during iterations. Specifically, LearnBD includes three phases: cut sampling (Phase 1), classifier construction (Phase 2), and cut classification (Phase 3). We show the algorithm overview in Figure 1 in which related to Phase 1, \( K \) represents the number of sampling paths, \( N \) represents the length of each sampling path, \( \text{RMP}_n^k, n = 1, \ldots, N, k = 1, \ldots, K \) is the RMP corresponding to iteration \( n \) in sampling path \( k \). Related to Phase 3, \( \text{RMP}_t^t, t = 0, \ldots, T \) is the RMP corresponding to the iteration \( t \) and \( t = T \) denotes the last iteration in Phase 3. In Phase 1, we perform cut sampling to generate training data, including cut characteristics and performance measures (see the details in Section 3.1). In Phase 2, we train an SVM classifier with the training data generated in Phase 1, which takes cut characteristics as input and \( \{1, -1\} \) valued label as output to classify whether or not a cut is valuable (see Section 3.2). In Phase 3, we utilize the classifier to distinguish valuable cuts with label \( = 1 \) from all generated cuts in each iteration and solve RMPs iteratively by only adding valuable cuts (see Section 3.3).

3.1. Phase 1: Cut Sampling

In Phase 1, we conduct cut sampling to collect the information of valuable cuts, which will then be used to train the classifier in Phase 2. The training data set \( D \) can be viewed as a \( D_{\text{row}} \times D_{\text{col}} \)}
matrix, where each row is the information of a specific sample cut, the first $D_{col} - 1$ columns are cut characteristics, and the last column is a $\{-1, 1\}$ valued label.

**Characteristics.** Cut characteristics are features of a cut that can help us predict performance of the cut in future iterations and they are collected before adding the cut to RMP. We consider the following two characteristics. The first is cut violation at the current solution $(\hat{x}_t, \hat{\theta}_t)$ of $\text{RMP}_t$, which is denoted by $VL$ and can be computed as $\pi^T_T(h_\omega - T_\omega \hat{x}_t) - \hat{\theta}_t$, according to (8). This characteristic reflects how large the feasible region of $\text{RMP}_t$ can be cut off if adding the cut. The second characteristic is related to the scenario where a cut is generated from. We denote the number of cuts generated by the same scenario in previous iterations as $NC$. This characteristic reflects the trade-off between exploration and exploitation, two typical learning strategies. A preference to a cut whose associated scenario generates more cuts in previous iterations, links to an exploitation strategy, while the opposite preference leads to an exploration strategy. On the one hand, a large number of cuts generated from the same scenario shows that this scenario is crucial for identifying an optimal solution. However, it could be the case where the majority of valuable cuts of this scenario have been already generated and thus the change of the objective value of $\text{RMP}$ brought by a new cut from the same scenario is small. Therefore, the relationship between $NC$ and future performance of a cut is highly possible to be nonlinear. A collection of characteristics of one specific cut is referred to as an observation $o$, with $o = \{VL, NC\}$. 

![Figure 1](https://example.com/figure1.png)  
**Figure 1** Overview of LearnBD.
Label and performance index. In training data set, each observation also needs to be assigned a label \( l \), where 1 is assigned to valuable cuts and \(-1\) is assigned to non-valuable cuts. Therefore, we define a performance index of each cut and then transform it into \( \{-1, 1\} \) valued label. We choose the change amount of the objective value of RMP\(^t\) before and after adding a cut as the performance index of the cut, denoted by PI. We add exactly one cut to RMP\(^t\) each time to recognize the change of objective value brought by the cut. In practice, users can customize the characteristics and performance index according to specific applications. The rule for transforming the performance index will be discussed after we introduce sampling paths next.

Sampling path. We construct sampling paths to guide the cut sampling process to record the cut characteristics and performance index. The number of sampling paths \( K \) and the length of sampling path \( N \) are pre-determined hyperparameters. In each sampling path \( k \), \( k = 1, \ldots, K \), we start with RMP\(_k^0\), which is initialized by RMP\(^0\). In iteration \( n \) of a sampling path \( k \), for \( n = 0, \ldots, N-1, k = 1, \ldots, K \), we solve RMP\(_k^n\) and obtain an optimal solution \((\hat{x}_k^n, \hat{\theta}_k^n)\). Then we follow the Monte Carlo sampling approach to randomly sample one scenario \( \omega \in \Omega \) and solve the corresponding SP\(_\omega\) by plugging in \((\hat{x}_k^n, \hat{\theta}_k^n)\): (i) if no optimality cut is generated, then we continue sampling another scenario and solving the corresponding SP; (ii) if an optimality cut is generated, we record the two characteristics, instantly add the cut to RMP\(_k^{n+1}\), and then record the performance index. Similar to Section 1.1, we use \( V_{\omega,n}^k \) to denote the set of identified extreme points of the feasible region of SP\(_\omega\) in iteration \( n \) of sampling path \( k \), and RMP\(_k^n\) is defined in the following form:

\[
(RMP_k^n) \min_{x \in X, \theta} \ c^T x + \sum_{\omega \in \Omega} p_\omega \theta_\omega \\
\text{s.t.} \quad \theta_\omega \geq (h_\omega - T_\omega x)^T \nu_\omega, \quad \omega \in \Omega, \ \nu_\omega \in V_{\omega,n}^k.
\]

(10)

Once a new cut generated, we move one step forward in one sampling path and therefore the iteration process stops after reaching RMP\(_k^N\) in sampling path \( k \), \( k = 1, \ldots, K \).

Remark 1. The cut sampling process is independent across all sample paths, and thus the cut information collected in different sampling paths is independent from each other.

Through cut sampling, we collect \( \Gamma = N \times K \) number of training data. Generally speaking, a larger set of training data leads to a more precise classifier. Larger \( N \) means that we can collect information of more representative cuts because we solve RMPs in a wider range of sizes. However, larger \( N \) also results in a longer data collection process in each sampling path and may result in longer solving time in total. Different independent sampling paths can be conducted in parallel, and therefore, larger \( K \) will not significantly increase the time of Phase 1. However, the cuts generated by RMPs with similar sizes can share similar characteristics. More similar cuts can also lead to over-fitting and can eventually weaken the power of the approach. Typically, we have \( \Gamma < 2|\Omega|\).
Label transformation. We define a label transformation function to transform continuous performance index into \{-1,1\} label, where 1 indicates a valuable cut. Benders decomposition’s convergence nature is accompanied by the fact that the change of the objective value of RMPs is decreasing over iterations. Therefore, we treat the cuts that can bring a large enough proportion of PI of the next cut in the same sampling path as valuable cuts. We directly assign label 1 to the last cut of each cut sampling path. For other cuts, we calculate the ratio of its PI and the PI of the next cut in the same sampling path and then compare the ratio with a predetermined threshold $\Delta \in [0,2]$. The label transformation function is defined as:

$$l_k^n = \begin{cases} -1, & \text{if } \frac{\text{PI}_k^n}{\text{PI}_{k+1}^n} < \Delta, \\ 1, & \text{otherwise,} \end{cases}$$

(11)

Larger $\Delta$ shows a more strict rule for recognizing a cut as a valuable cut. With this label transformation function, we can calculate all labels by current performance indices and then we obtain a training data set $D$ with $\Gamma$ rows and three columns, such that $D = \{(o_d, l_d), \; d = 1, \ldots, \Gamma\}$. We present the algorithmic details in Algorithm 1.

Remark 2. Label transformation function eliminates a degree of dependency across cuts generated in the same sampling path. Together with Remark 1, all training data are independent with each other.

3.2. Phase 2: Classifier Construction

In Phase 2, we construct an SVM classifier with training data $D$ to predict the potential performance of cuts and identify valuable cuts. As mentioned in Section 3.1, $D$ can be presented as a collection of observations and labels of sample cuts, where $D = \{(o_d, l_d), \; d = 1, \ldots, \Gamma\}$ and more specifically each cut observation $o = (VL, NC)$. In Section 3.2.1, we show how SVM works and how to estimate the parameters with training data. In Section 3.2.2, we discuss the advantages of SVM as a cut classifier.

3.2.1. Building Cut Classifier Using Support Vector Machines (SVM)

SVM is a well-known supervised machine learning approach (see Cortes and Vapnik, 1995; Vapnik, 1998, 1999, 2013) and has been used for analyzing data in many applications. Given a training data set $D = \{(o_d, l_d), \; d = 1, \ldots, \Gamma\}$ from Phase 1, where $o_d \in \mathbb{R}^{\Sigma}$ is an observation (in our problem $\Sigma = 2$), a subset of training data are identified as support vectors after the training process. Parameterized by a coefficient vector $a \in \mathbb{R}^\Gamma$, and an intercept $b \in \mathbb{R}$, the SVM classifier $f_{SVM}(\cdot) : \mathbb{R}^{\Sigma} \rightarrow \{-1,1\}$ for a new observation $o' \in \mathbb{R}^{\Sigma}$ (i.e., the collected information of a specific cut in our problem), is given by

$$f_{SVM}(o') = \text{sign} \left[ \sum_{d=1}^{\Gamma} l_da_dK(o', o_d) + b \right],$$

(12)
Algorithm 1 Phase 1 of the LearnBD algorithm.

1: **Input:** a two-stage stochastic program with a set Ω of scenarios; values of $N$, $K$, $Δ$.
2: **Initialize:** $\text{RMP}^0$, $\text{SP}_ω$, $ω ∈ Ω$, $D ← ϕ$.
3: for $k = 1, \ldots, K$ do
   4: **Initialize:** $V^0_ω$, $ω ∈ Ω$, $NC_ω = 0$, $\text{RMP}^0_k ← \text{RMP}^0$, $D_{\text{temp}} ← ϕ$.
   5: Solve $\text{RMP}^0_k$, obtain an optimal solution $\{\hat{x}_k^0, \hat{θ}_k^0\}$ with optimal objective value $\hat{z}_k^0$.
   6: for $n = 0, \ldots, N − 1$ do
      7: Randomly select $ω' ∈ Ω$, solve $\text{SP}_ω'$, obtain an optimal solution $\pi_{ω'}$ and its objective value $ζ_{ω'}$.
      8: if $(\hat{θ}_n^k)_ω < ζ_{ω'}$ then
         9: $V^{n+1}_k ← V^n_k ∪ \{\pi_{ω'}\}$, $NC_{ω'} ← NC_{ω'} + 1$, $VL ← ζ_{ω'} − (\hat{θ}_n^k)_ω$;
      else
         Go to Step 7.
   end if
   10: Solve $\text{RMP}^{n+1}_k$, obtain an optimal solution $\{\hat{x}^{n+1}_k, \hat{θ}^{n+1}_k\}$ with optimal objective value $\hat{z}^{n+1}_k$,
   11: $\text{PI} ← |\hat{z}^{n+1}_k − \hat{z}^n_k|$, 
   12: $(D_{\text{temp}})^{n+1}_k ← (VL, NC_{ω'}, \text{PI})$.
   13: end for
   14: $D ← D ∪ ((D_{\text{temp}})^{N, 1}_k, (D_{\text{temp}})^{N, 2}_k, 1)$
   15: for $n = N − 1, \ldots, 1$ do
      16: if $(D_{\text{temp}})^{n, 3}_k < Δ$ then
         17: $D ← D ∪ ((D_{\text{temp}})^{n, 1}_k, (D_{\text{temp}})^{n, 2}_k, −1)$
      else
         18: $D ← D ∪ ((D_{\text{temp}})^{n, 1}_k, (D_{\text{temp}})^{n, 2}_k, 1)$
      end if
   end for
   19: $k^* = \arg \min_{k = 1, \ldots, K} \hat{z}^N_k$
   20: return $D$, $\text{RMP}^N_k$.

where $K(\cdot, \cdot): \mathbb{R}^Ω × \mathbb{R}^Ω → \mathbb{R}$ is a predetermined kernel function. One famous kernel function is the Radial Basis Function (RBF), in which $K(o_1, o_2) = \exp(γ∥o_1 · o_2∥^2)$.

The label prediction function $f_{SV,M}(\cdot)$ can be interpreted as follows. We can treat the coefficient $a_d$ as a significance-magnitude of the corresponding data point $d = 1, \ldots, Γ$, because the label $l_d$ is always shown in $a_d × l_d$ in the predicting process and $a_d × l_d$ as a whole indicates the power of data.
point $d$ for classifying new cuts. The kernel function $K(o', o_d)$ presents the similarity between the characteristics of a new cut $o'$ and cut $o_d$. Then the label of $o'$ is the sign of a sum of magnitude-adjusted label of all training data points plus an intercept $b$. Then by eliminating the training data with zero estimated coefficients $a_d$, the remaining training data form the support vector set $S$, which is a subset of $D$, and function (12) can be simplified as

$$f_{SVM}(o') = \text{sign} \left[ \sum_{s \in S} l_s a_s K(o', o_s) + b \right]. \quad (13)$$

The parameters $\{S, a, b \}$ can be trained by minimizing the prediction loss function among training data as well as maximizing the flatness of the boundary between valuable and non-valuable cuts. The prediction loss is computed by hinge loss function to improve the model sparsity. Given the estimated result $u = \sum_{d=1}^{\Gamma} l_d a_d K(o', o_d) + b$ from Equation (12) and the ground truth label $l$, the loss is calculated by:

$$\text{Loss}(u, l) = \max(0, 1 - l \cdot u).$$

It can be seen that when $l$ and $u$ have the same sign and $|u| \geq 1$, the loss $= 0$; otherwise the loss $= |u - l|$. 

To elaborate the training process of SVM, we start with

$$f_{SVM}(o') = \text{sign} \left[ w^T \phi(o') + b \right] \quad (14)$$

and $u = w^T \phi(o') + b$, where $\phi(\cdot)$ is a unique mapping function such that $K(o_1, o_2) = \langle \phi(o_1), \phi(o_2) \rangle$. By the kernel trick (see, e.g., Schölkopf et al., 2002), we do not have to know the exact form of $\phi(\cdot)$ and we can employ the SVM model only with kernel function $K(\cdot, \cdot)$. In Proposition 4, we show this formulation is equivalent to (13). With a penalty hyperparameter $C \geq 0$ assigned to the prediction error, the objective function for solving the parameters is defined as $\frac{1}{2} w^T w + C \sum_{d=1}^{\Gamma} \xi_d$, where the first term representing the flatness and $\xi_d$, $d = 1, \ldots, \Gamma$ is an auxiliary variable for representing loss amount of training data $(o_d, l_d)$. The parameters can be solved by an optimization problem:

$$(\text{SVM-P}) \quad \min_{w, \xi, b} \frac{1}{2} w^T w + C \sum_{d=1}^{\Gamma} \xi_d \quad (15a)$$

s.t. $l_d \cdot (w^T \phi(o_d) + b) \geq 1 - \xi_d \quad d = 1, \ldots, \Gamma; \quad (15b)$

$$\xi_d \geq 0 \quad d = 1, \ldots, \Gamma, \quad (15c)$$

where (15b) are used to calculate the hinge loss and (15c) are sign restrictions of $\xi$. (SVM-P) is a convex optimization problem with convex inequality constraints and a quadratic objective function, and thus it is easy to solve by taking Lagrangian Dual and applying Krash-Kuhn-Tucker (KKT) conditions (see, e.g., Chang and Lin, 2011).
Proposition 1. The optimal objective value of

$$\min_{w, \xi, b} \frac{1}{2} w^T w + C \sum_{d=1}^{\Gamma} \xi_d - \sum_{d=1}^{\Gamma} a_d \{ l_d \cdot (w^T \phi(o_d) + b) - 1 + \xi_d \} - \sum_{d=1}^{\Gamma} v_d \xi_d$$

(16)

with any $a, v \geq 0$ is a valid lower bound of (SVM-P).

Proof. Assume that $(w_1, \xi_1, b_1)$ is an optimal solution to (SVM-P), and therefore it is feasible to the relaxation [16]. Given the constraints in (SVM-P), we have $l_d \cdot (w_1^T \phi(o_d) + b_1) - 1 + \xi_{1d} \geq 0$ and $\xi_{1d} \geq 0$ for all $d$. Therefore, for any $a, v \geq 0$, the objective value of (16) based on solution $(w_1, \xi_1, b_1)$ is no larger than $\frac{1}{2} w_1^T w_1 + C \sum_{d=1}^{\Gamma} \xi_{1d}$. Moreover, as the optimal objective value of (16) is smaller than or equal to the objective value of any feasible solution, we can conclude that the objective value of (16) evaluated at the feasible solution $(w_1, \xi_1, b_1)$ is always smaller than or equal to $\frac{1}{2} w_1^T w_1 + C \sum_{d=1}^{\Gamma} \xi_{1d}$, which is the optimal objective value of (SVM-P) and thus provides a valid lower bound of (SVM-P). This completes the proof. □

By associating dual variables $a \geq 0$ with inequality constraints (15b) and dual variables $v \geq 0$ with inequality constraints (15c), we can relax those two sets of constraints and then obtain the corresponding Lagrangian function for any feasible solution $(w, \xi, b)$ as

$$L(w, \xi, b; a, v) = \frac{1}{2} w^T w + C \sum_{d=1}^{\Gamma} \xi_d - \sum_{d=1}^{\Gamma} a_d \{ l_d \cdot (w^T \phi(o_d) + b) - 1 + \xi_d \} - \sum_{d=1}^{\Gamma} v_d \xi_d.$$

By weak duality, the Lagrangian problem

$$\min_{w, \xi, b} L(w, \xi, b; a, v)$$

yields a valid lower bound of (SVM-P). Moreover,

$$\max_{a \geq 0, v \geq 0} \min_{w, \xi, b} L(w, \xi, b; a, v)$$

is the dual problem that seeks the best lower bound.

Definition 1. Krash-Kuhn-Tucker (KKT) is a set of conditions including: primal feasibility, dual feasibility, complementary slackness, and the first derivative of Lagrangian function $L(\cdot)$ being zero. If the primal problem is

$$\min_x f_0(x)$$

subject to $f_i(x) \leq 0 \ \forall i \in I$

$h_i(x) = 0 \ \forall i \in I'$

and the associated dual multipliers are $\lambda \geq 0$ and $\mu$, then the KKT conditions are:

- $f_i(x^*) \leq 0 \ \forall i \in I$ and $h_i(x^*) = 0 \ \forall i \in I'$ (primal feasibility),
- $\lambda^* \geq 0$ (dual feasibility),
• \( f_i(x^*)\lambda^*_i = 0 \) (complementary slackness),
• \( \nabla f_0(x^*) + \sum_{i \in I} \lambda^*_i \nabla f_i(x^*) + \sum_{i' \in I'} \mu^*_i \nabla h_i(x^*) = 0 \) (first derivative of \( L(\cdot) \) is zero).

**Proposition 2.** The strong duality holds for *(SVM-P)* and KKT conditions are satisfied at the optimal primal and dual solution pair.

**Proof.** *(SVM-P)* has a quadratic objective and affine inequality constraints, and therefore by Slater’s condition strong duality holds. Because *(SVM-P)* is differentiable, KKT conditions hold at the global optimum. This completes the proof. □

**Theorem 1.** The optimal objective value of the optimization problem

\[
\max_{a \geq 0, v \geq 0} \min_{w, \xi, b} \frac{1}{2} w^T w + C \sum_{d=1}^r \xi_d - \sum_{d=1}^r a_d \left\{ l_d \cdot (w^T \phi(o_d) + b) - 1 + \xi_d \right\} - \sum_{d=1}^r v_d \xi_d \tag{17}
\]

equals to the optimal objective value of *(SVM-P)*.

**Proof.** Recall the Lagrangian dual problem

\[
\max_{a \geq 0, v \geq 0} \min_{w, \xi, b} L(w, \xi, b; a, v).
\]

By Proposition 2 strong duality holds and thus the optimal objective value of the dual problem and primal problem are equal. □

**Proposition 3.** The Lagrangian dual function *(16)* in Proposition 1 can be reformulated as

\[
\frac{1}{2} \sum_{d=1}^r \sum_{d' = 1}^r l_d l_{d'} a_d a_{d'} K(o_d, o_{d'}) + \sum_{d=1}^r a_d - \sum_{d=1}^r v_d \xi_d \tag{18a}
\]

with \( \sum_{d=1}^r a_d l_d = 0; \) \( C - a_d - v_d = 0 \quad d = 1, \ldots, \Gamma. \) \( \tag{18b} \)

**Proof.** The Lagrangian dual function *(16)* is differentiable, and therefore the derivatives associated with \((w, \xi, b)\) at the minimum are equal to zero, i.e.,

\[
\frac{\partial L}{\partial w} = 0 \rightarrow w = \sum_{d=1}^r a_d l_d \phi_d \tag{19a}
\]

\[
\frac{\partial L}{\partial b} = 0 \rightarrow \sum_{d=1}^r a_d l_d = 0 \tag{19b}
\]

\[
\frac{\partial L}{\partial \xi} = 0 \rightarrow C - a_d - v_d = 0, \quad d = 1, \ldots, \Gamma \rightarrow a_d \leq c, \quad d = 1, \ldots, \Gamma. \tag{19c}
\]

Plugging in the results in *(19)*, we can obtain the reformulation of *(16)* in *(18)*. This completes our proof. □
Theorem 2. The Lagrangian dual problem (17) is equivalent to solving a convex quadratic program:

$$\max_a \frac{1}{2} \sum_{d=1}^{r} \sum_{d'=1}^{r} l_{dd'} a_d a_{d'} K(o_d, o_{d'}) + \sum_{d=1}^{r} a_d \quad (20a)$$

s.t. \( \sum_{d=1}^{r} a_d l_d = 0; \quad (20b) \)

\( 0 \leq a_d \leq C \quad d = 1, \ldots, \Gamma. \) \quad (20c)

Proof. By Proposition 3, we obtain an equivalent formulation of (17) as follows.

$$\max_{a, \xi \geq 0} \frac{1}{2} \sum_{d=1}^{r} \sum_{d'=1}^{r} l_{dd'} a_d a_{d'} K(o_d, o_{d'}) + \sum_{d=1}^{r} a_d - \sum_{d=1}^{r} v_d \xi_d \quad (21a)$$

with \( \sum_{d=1}^{r} a_d l_d = 0; \quad (21b) \)

\( C - a_d - v_d = 0 \quad d = 1, \ldots, \Gamma; \quad (21c) \)

In the third term in the objective function (21a), all \( v_d \xi_d, \forall d = 1, \ldots, \Gamma \) are zero at the optimum because of the complementary slackness by Proposition 2. Therefore, we can discard the third term without loss of optimality. Moreover, because \( v_d \geq 0, \forall d = 1, \ldots, \Gamma \), we can combine (21c) with \( v \geq 0 \) and derive valid constraints \( a_d \leq C, \forall d = 1, \ldots, \Gamma \), which helps to eliminate variables \( v_d, \forall d = 1, \ldots, \Gamma \). Finally, we can rewrite model (21) as shown in (20) (see, e.g., Chang and Lin 2011). This completes our proof.

Proposition 4. The parameter of the classifier in (14) are \( w^* = \sum_{d=1}^{r} a_d^* l_d \phi_d \) and \( b^* = 1 - \sum_{d'=1}^{r} l_{d'}(a_{d'}^* K(o_d, o_{d'})) \) for any \( d = 1, \ldots, \Gamma \), associated with \( a_d^* \in (0, C) \). The three prediction functions (12), (13) and (14) are equivalent to each other, where the support vector set \( S \) in (13) contains all \( (o_d, l_d) \), \( d = 1, \ldots, \Gamma \) such that \( a_d^* > 0 \).

Proof. The value of \( w^* \) is obtained by (19a) and it shows the equivalence between (12) and (14). Assume that we solve and obtain an optimal solution \( a^* \) to (20). Then following the complementary slackness:

\( a_d \left[ l_d \cdot (w^T \phi(o_d) + b) \right] - 1 + \xi_d = 0, \forall d = 1, \ldots, \Gamma, \)

we have:

- If \( a_d^* = C > 0 \), then \( l_d \cdot (w^* T \phi(o_d) + b) = 1 - \xi_d^* \). By \( a_d^* = C - v_d^* \) we have \( v_d^* = 0 \) and thus \( \xi_d^* \geq 0 \).

The observation \( d \) is called non-margin support vector.

- If \( 0 < a_d^* < C \), then \( l_d \cdot (w^* T \phi(o_d) + b) = 1 - \xi_d^* \). Similarly, we have \( v_d^* > 0 \) and thus \( \xi_d^* = 0 \). The observation \( d \) is called margin support vector. Therefore, we can compute \( b^* = 1 - \sum_{d'=1}^{r} l_{d'}(a_{d'}^* K(o_d, o_{d'})) \) with any \( d = 1, \ldots, \Gamma \) associated with \( a_d^* \in (0, C) \),
• If $a_d^* = 0$, then this type of observation $d$ does not affect the value of the second prediction function. Therefore, we can build a support vector set $S$ of $(o_d, l_d)$, $d = 1, \ldots, \Gamma$ with $a_d^* \neq 0$ and thus simplify (12) as (13).

\[ \square \]

**Remark 3.** The penalty hyperparameter $C$ balances the explanatory and predictive power of the classifier. In general, a larger $C$ shows a smaller tolerance of prediction error within training dataset and hence results in a classifier with higher explanatory power while too large $C$ will destroy the predictive power. The discount rate $\gamma$ in RBF kernel determines the magnitude of similarity between observations, which is related to the model sensitivity and convergence property. Proper $(\gamma, C)$ will generate a relatively small number of support vectors to improve prediction efficiency without losing main predictive power. Those two hyperparameters are generally selected together via cross-validation and grid search to reach the best empirical performance.

**3.2.2. Reasons for Choosing SVM** The advantage of using support vector type of methods for cut classification is threefold. Firstly, with the help of hinge loss function, SVM only selects representative observations from the training data. Those observations are referred to as support vectors and are stored for future classification. This sparsity nature increases the computational speed for evaluating new cuts. Secondly, the mechanism of SVM can be explained by using the similarity between a new cut and all support vectors to predict the future performance, which is consistent with our assumptions and motivation that valuable cuts share similarities. Furthermore, the kernel-based method can flexibly help capture the nonlinear relationship between cut performance and characteristics. Thirdly, the solving process of SVM is a convex optimization problem (see model (SVM-P) in (15)) which is computationally tractable.

Another support vector type of learning method is support vector regression (SVR) (see, e.g. Smola 2004). The main idea of those two approaches is similar. SVM classifies cuts by \{1, -1\} labels and works as a classifier. SVR evaluates continuous scores of cuts and works as a regressor. A regressor is more informative compared with a classifier because it can distinguish more rank levels and also allows any fractional rank between levels. We choose SVM over SVR following concerns listed as follows. Indeed, cuts have different levels of effectiveness for improving RMP solutions. However, we choose not to spend time and effort to fully distinguish between those levels. Recall that the geometric explanation of Benders decomposition is to cut off the feasible region of RMP$^t$ in each iteration $t$. The cuts generated in one iteration can be linearly independent with each other and thus they cut the feasible region from different directions. Therefore, it is better to include several valuable cuts rather than only one cut in each iteration. On the other hand, since we do not select the cuts with relatively low effectiveness, we even do not need to distinguish among those non-valuable cuts. A similar reason is also mentioned in the learning approach used for a branch-and-bound algorithm by Khalil et al. (2016).
3.3. Phase 3: Cut Classification

Iteration rule. In Phase 3, the algorithm starts with $\text{RMP}^0$ in iteration 0 and we later elaborate the initialization options in the next paragraph. In iteration $t$, we solve $\text{RMP}^t$ and obtain an optimal solution $(\hat{x}^t, \hat{\theta}^t)$; by plugging in $(\hat{x}^t, \hat{\theta}^t)$, we solve all $\text{SP}_{\omega}$, $\omega \in \Omega$ and record the two characteristics of each generated cut. Using the characteristic information, the SVM classifier assigns label 1 to valuable cuts and $-1$ to non-valuable cuts. Then we add all valuable cuts with label = 1 to $\text{RMP}^{t+1}$. We use $\mathcal{V}_{\omega,t}$ to denote the identified extreme points of the feasible region of $\text{SP}_{\omega}$ in iteration $t$ in Phase 3. We repeat the above process until the optimality gap between upper bound and lower bound, which is defined in (9), is less than a pre-specified tolerance $\delta$. If no cut is labeled 1 by SVM classifier but we have not reached the optimal tolerance, we switch to traditional Benders decomposition. We present the algorithmic details in Phase 3 in Algorithm 2.

Initialization options. The $\text{RMP}^0$ has two initialization options. The first option is $\text{RMP}_{k^*}^N$, where $k^* = \arg \max_{k=0, \ldots, K} \hat{z}_k^N$ and $\hat{z}_k^N, k = 0, \ldots, K$ is the objective value of $\text{RMP}_k^N$ (also shown in Algorithm 1). The second option is $\text{RMP}^0$, defined in (3). We also initialize the identified extreme points of scenarios $\mathcal{V}_{\omega,0}^0$, $\forall \omega \in \Omega$ (to reflect the identified cuts) and the number of added cuts of each scenario $\text{NC}_{\omega} = |\mathcal{V}_{\omega,0}^0|$, $\forall \omega \in \Omega$. The optimal objective values of RMPs increase over iterations, and thus $\text{RMP}_{k^*}^N$ with the highest objective value is the closest to the true optimal solution. Choosing $\text{RMP}_{k^*}^N$ will result in fewer iterations in Phase 3. However, $\text{RMP}_{k^*}^N$ may include non-valuable cuts because the existing cuts are selected randomly during Phase 1. Re-solving RMPs with those non-valuable cuts in Phase 3 are computationally inefficient. Instead, choosing $\text{RMP}^0$ ensures that $\text{RMP}^t$ only contains valuable cuts which are classified by the SVM classifier and thus it can potentially reduce solving time in future iterations. In general, $\text{RMP}_{k^*}^N$ helps to reduce the number of iterations and $\text{RMP}^0$ leads to shorter computing time in each iteration.

4. Numerical Studies

We consider the capacitated facility location problem (CFLP) to generate instances for verifying the performance of the LearnBD algorithm. The well-known facility location problem (see, e.g. Geoffrion and Graves, 1974 [Birge and Louveaux, 2011]) is an NP-hard combinatorial optimization problem and can be formulated as a two-stage stochastic programming problem with a finite scenario set $\Omega$ of different demand realizations. Consider a set $W$ of production plants (facilities) and a set $F$ of factories which have uncertain demand $\tilde{d}$. The setup cost of facility $i \in W$ is $k_i$ and the production capacity limit is $u_i$. The demand of factory $j \in F$ can be satisfied by products produced in open facility $i \in W$ with an unit transportation cost $c_{ij}$, and the unsatisfied demand will generate lost-sale with a unit penalty cost $\rho_j$. The two-stage stochastic programming model consists of two types of decisions. In the first stage, we make decisions before realizing the demand
values. We define binary decision variables $x_i, \forall i \in W$ such that $x_i = 1$ if we open facility $i$ and $x_i = 0$ otherwise. In the second stage, we obtain the demand value from each factory and define continuous decision variables $y_{ij} \geq 0, \forall i \in W, j \in F$, which represent transportation units from facility $i$ to factory $j$. The model aims to find the best decisions to minimize the facility setup cost, expected transportation cost, and expected lost-sale cost. The first-stage formulation is:

$$
(CFLP) \quad \min_x \sum_{i \in W} k_i x_i + \sum_{\omega \in \Omega} p_\omega Q_\omega(x)
\text{ s.t. } x_i \in \{0,1\} \quad i \in W.
$$

(22)
The second-stage problem for each scenario \( \omega \) is defined with transportation decision variables \( y_{ij}, \ i \in W, \ j \in F \) and auxiliary variables \( \alpha_j, \ j \in F \) for unsatisfied demand:

\[
Q_\omega(x) = \min_{y, \alpha} \sum_{i \in W} \sum_{j \in F} c_{ij} y_{ij} + \sum_{j \in F} \rho_j \alpha_j \\
\text{s.t.} \sum_{j \in F} y_{ij} \leq u_i x_i \quad i \in W; \\
\tilde{d}_{\omega,j} - \sum_{i \in W} y_{ij} \leq \alpha_j \quad j \in F; \\
y_{ij} \geq 0 \quad i \in W, \ j \in F; \\
\alpha_j \geq 0 \quad j \in F.
\]

(23)

With the tolerance of unsatisfied demand, the problem always has a feasible solution and Benders decomposition only generates optimality cuts. To apply the traditional Benders approach and LearnBD algorithm, we derive the dual of second-stage problems and formulate SPs as shown in Section 1.1. By defining dual variables \( h_i, \ \forall i \in W \) and \( \pi_j, \ \forall j \in F \), respectively associated with the first and second constraints in model (23), we formulate the subproblem in scenario \( \omega \) as

\[
(SP_\omega) \max_{h, \pi} - \sum_{i \in W} u_i x_i h_i + \sum_{j \in F} \tilde{d}_{\omega,j} \pi_j \\
\text{s.t.} \quad -h_i - \pi_j \leq c_{ij} \quad i \in W, \ j \in F; \\
0 \leq \pi_j \leq \rho \quad j \in F; \\
h_i \geq 0 \quad i \in W.
\]

(24)

To specify model (5), letting \( V^{\omega,t} \) be a collection of extreme points of \( SP_\omega \) that have been identified when reaching iteration \( t \), we formulate

\[
(RMP^t) \min_{x, \theta} \sum_{i \in W} k_i x_i + \sum_{\omega \in \Omega} p_{\omega} \theta_\omega \\
\text{s.t.} \quad \theta_\omega \geq - \sum_{i \in W} u_i x_i h_i + \sum_{j \in F} \tilde{d}_{\omega,j} \pi_j \quad (h_j, \pi_j) \in V^{\omega,t}, \ \omega \in \Omega; \\
x_i \in \{0,1\} \quad i \in W.
\]

(25)

We test instances involving 5 to 20 facility location candidates and 4 to 50 factories. The capacity limit and setup cost of each facility location candidate are sampled from a uniform distribution between \([200 \times \text{number of factories}, \text{number of facilities}] \times [100, \text{1000}] \), respectively. The lost sale penalty cost and travel cost are sampled from a uniform distribution between \([10, 25] \) and \([1, 5] \), respectively. The uncertain demand of each factory follows a truncated normal distribution, with mean sampled from uniform distributions between \([100, 300] \) and standard deviation from a uniform distribution between \([5, 25] \). Hyperparameters \((K, N, \Delta)\) in Phase 1 are specified later in each instance and \((C = 1, \gamma = \frac{1}{2})\) in Phase 2. All the tests are performed on a computer with an Intel Core E5-2630 v4 CPU 2.20 GHz and 128 GB of RAM. To compare the performance between the traditional Benders approach and the LearnBD algorithm, we present results over iterations for two instances in Section 4.1 and also statistical results in Section 4.2.
4.1. Performance Comparison over Iterations

To track the performance of the traditional Benders approach and LearnBD, we present the results of two specific instances. For each instance, we solve its replications with both traditional Benders and LearnBD separately to the same optimality gap, where in LearnBD, the Phase 3 begins with RMP\(^0\). For each approach, we record the following values after each iteration:

- the optimality gap after the current iteration, which helps record the algorithm convergence;
- the cumulative time for solving RMPs, which is the main bottleneck of the algorithm efficiency;
- the total number of cuts added to RMPs in the previous iterations, which reflects the size of the RMPs and power of the classifier.
- the cumulative time for solving SPs, which can be performed in parallel in each iteration and therefore this value will not affect the algorithm efficiency.

We first consider an instance with 5 facility candidates, 4 factories, and 3000 uncertain scenarios. We solve it until \(\delta = 0.1\%\), and for LearnBD we choose parameters \(K = 2\), \(N = 3000\), and \(\Delta = 1\). The results over the iterations are shown in Figure 2. LearnBD requires 6 iterations while the traditional Benders approach can solve the instance in 3 iterations. LearnBD only adds a much smaller number of cuts but achieves the similar optimality gap and higher ratio of tight cuts, especially in the first three iterations, which indicates the power of our SVM cuts classifier. Due to the smaller sizes of RMPs, the cumulative time for solving RMPs of LearnBD is remarkably shorter than that of traditional Benders approach, although the former takes more iterations. Because the SPs can be solved in parallel, we only present the cumulative time for solving SPs.

Second, we consider an instance with 10 facility candidates, 20 factories, and 1000 uncertain scenarios. We solve it until \(\delta = 1\%\), and for LearnBD we choose parameters \(N = 2\), \(N = 1000\), \(\Delta = 1.005\). The results over the iterations are similar to the ones above and are shown in Figure 3. LearnBD takes more iterations (37 iterations) than the traditional Benders approach (28 iterations). However, LearnBD adds fewer cuts and achieves the similar optimality gap. Especially in the first several iterations, the cumulative time for solving RMPs of LearnBD is half of that of the traditional Benders approach.

4.2. Performance Comparison in Statistics

In this section, we present the statistical performance comparison between the results given by the traditional Benders approach and LearnBD. We test several instances with different number of facilities, factories, and scenarios. For each instance, we randomly generate the data for ten replications, according to the generating mechanism stated before, and record the same four values as Section 4.1 (i.e., the optimality gap, the cumulative time for solving RMPs, the total number of cuts, and the cumulative time for solving SPs) after the last iteration. For each value, we calculate
the sample mean, standard deviation, maximum and minimum. Here we include instances with the same data settings in Section 4.1.

We first consider an instance with 5 facility candidates, 4 factories, and 3000 uncertain scenarios. We solve it until $\delta = 0.1\%$, and for LearnBD we choose parameters $K = 2$, $N = 3000$, $\Delta = 1$. The results are shown in Table 1. We observe that LearnBD reduces the number of Benders cuts and improves the efficiency for solving two-stage stochastic programs significantly. The solving time of RMPs is reduced by 61.4% in instances on average.

Second, we consider an instance with 10 facility candidates, 20 factories, and 1000 uncertain scenarios. We solve it until $\delta = 1\%$, and for LearnBD we choose parameters $K = 2$, $N = 1000$, $\Delta = 1.005$. The results are shown in Table 2. In this instance, we can observe that the solving time of RMPs is reduced by 48.12% in instances on average.

5. Conclusions
In this paper, we develop a learning-enhanced algorithm to accelerate the solving process of Benders decomposition, one of the most useful algorithms for solving two-stage stochastic programs. The
Figure 3  Instance 2 solved by Benders decomposition (denoted by BD) and LearnBD. The horizontal axis is iteration number.

Table 1  Statistical result of ten tests for instance #1.

| Performance | Method | # of iter. | # of cuts | ratio of TC | Time-RMP (s) | Time-SP (s) |
|-------------|--------|------------|-----------|-------------|--------------|--------------|
| Average BD  | 3      | 9000.00    | 0.33      | 27.73       | 4.44         |
|             | LearnBD| 6          | 6884.60   | 0.44        | 10.70        | 9.52         |
| Standard BD | 0      | 0.00       | 0.00      | 9.81        | 1.46         |
|             | LearnBD| 0          | 98.24     | 0.01        | 2.91         | 2.97         |
| Maximum BD  | 3      | 9000.00    | 0.33      | 55.38       | 7.96         |
|             | LearnBD| 6          | 7033.00   | 0.45        | 18.23        | 17.83        |
| Minimum BD  | 3      | 9000.00    | 0.33      | 11.94       | 2.70         |
|             | LearnBD| 6          | 6701.00   | 0.43        | 5.13         | 6.01         |

bottleneck for traditional Benders decomposition is the increasing size of RMPs and the long solving time of RMPs, and our main goal is to restrict the sizes of RMPs by distinguishing valuable cuts. The computational studies based on capacitated facility location instances show the power of SVM cut classifier, and with a proper selection of hyperparameters, the LearnBD algorithm
Table 2  Statistical result of ten tests for instance #2.

| Performance | Method   | # of iter. | # of cuts | ratio of TC | Time-RMP (s) | Time-SP (s) |
|-------------|----------|------------|-----------|-------------|--------------|-------------|
| Average     | BD       | 41.20      | 41199.50  | 0.03        | 1772.92      | 23.76       |
|             | LearnBD  | 48.20      | 31786.10  | 0.04        | 1092.20      | 25.84       |
| Standard    | BD       | 48.20      | 31786.10  | 0.04        | 1092.20      | 25.84       |
| Derivation  | LearnBD  | 12.49      | 16097.65  | 0.01        | 1241.83      | 7.49        |
| Maximum     | BD       | 32.00      | 32000.00  | 0.02        | 616.14       | 16.16       |
|             | LearnBD  | 39.00      | 20035.00  | 0.02        | 249.17       | 19.64       |
| Minimum     | BD       | 63.00      | 62999.00  | 0.03        | 5179.20      | 50.17       |
|             | LearnBD  | 69.00      | 62356.00  | 0.05        | 3688.56      | 42.01       |

works efficiently with shorter solving time of RMPs and a higher ratio of tight cuts as compared to traditional Benders decomposition.

We attempted to build a learning-enhanced algorithm and achieved encouraging preliminary results. There are still several improvements can be made in the future. The first is to expend the characteristics and performance indices for the current LearnBD algorithm to capture multiple types of information of cuts. The second is to explore the possibility for constructing an online learning algorithm with the application of the reinforcement learning, which requires to decompose the effects of multiple cuts added simultaneously into the same RMP. For future research, we will also investigate improvements of LearnBD in a broader range of problems with special structural properties.

Acknowledgments

The authors gratefully acknowledge the support from the U.S. Department of Engineering (DoE) grant # de-sc0018018.

References

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

S. Binato, M. V. F. Pereira, and S. Granville. A new Benders decomposition approach to solve power transmission network design problems. *IEEE Transactions on Power Systems*, 16(2): 235–240, 2001.

J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer, 2011.

X. Cai, D. C. McKinney, L. S. Lasdon, and D. W. Watkins Jr. Solving large nonconvex water resources management models using generalized Benders decomposition. *Operations Research*, 49(2):235–245, 2001.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–27, 2011.
J.-F. Cordeau, G. Stojković, F. Soumis, and J. Desrosiers. Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, 35(4):375–388, 2001.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

A. M. Costa. A survey on Benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research*, 32(6):1429–1450, 2005.

T. G. Crainic, M. Hewitt, and W. Rei. Partial decomposition strategies for two-stage stochastic integer programs. Technical Report CIRRELT-2014-13, CIRRELT, Montreal, Canada, 2014.

A. Federgruen and P. Zipkin. A combined vehicle routing and inventory allocation problem. *Operations Research*, 32(5):1019–1037, 1984.

B. Gendron, M. G. Scutellà, R. G. Garropp, G. Nencioni, and L. Tavanti. A branch-and-Benders-cut method for nonlinear power design in green wireless local area networks. *European Journal of Operational Research*, 255(1):151–162, 2016.

A. M. Geoffrion and G. W. Graves. Multicommodity distribution system design by Benders decomposition. *Management Science*, 20(5):822–844, 1974.

H. He, H. Daume III, and J. M. Eisner. Learning to search in branch and bound algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3293–3301, 2014.

K. Holmberg. On the convergence of cross decomposition. *Mathematical Programming*, 47(1-3):269–296, 1990.

J. N. Hooker. Planning and scheduling by logic-based Benders decomposition. *Operations Research*, 55(3):588–602, 2007.

E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6348–6358, 2017.

E. B. Khalil, P. Le Bodic, L. Song, G. L. Nemhauser, and B. N. Dilkina. Learning to branch in mixed integer programming. In *Proceedings of the thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 724–731, 2016.

A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

M. Kruber, M. E. Lübbecke, and A. Parmentier. Learning when to use a decomposition. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 202–210. Springer, 2017.

G. Laporte, F. V. Louveaux, and H. Mercure. A priori optimization of the probabilistic traveling salesman problem. *Operations Research*, 42(3):543–549, 1994.

T. L. Magnanti and R. T. Wong. Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, 29(3):464–484, 1981.
M. Minoux. *Mathematical Programming: Theory and Algorithms*. John Wiley & Sons, 1986.

J. Naoum-Sawaya and S. Elhedhli. An interior-point Benders based branch-and-cut algorithm for mixed integer programs. *Annals of Operations Research*, 210(1):33–55, 2013.

W. Orchard-Hays et al. *Advanced Linear-programming Computing Techniques*. McGraw-Hill, 1968.

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

G. R. Raidl. Decomposition based hybrid metaheuristics. *European Journal of Operational Research*, 244(1):66–76, 2015.

B. Schölkopf, A. J. Smola, F. Bach, et al. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

A. J. Smola. *A Tutorial on Support Vector Regression*. Kluwer Academic Publishers, 2004.

V. Vapnik. *Statistical Learning Theory*. 1998, volume 3. Wiley, New York, 1998.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2013.

V. N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

P. Wolfe. Convergence theory in nonlinear programming. *Integer and Nonlinear Programming*, pages 1–36, 1970.

G. Zakeri, A. B. Philpott, and D. M. Ryan. Inexact cuts in Benders decomposition. *SIAM Journal on Optimization*, 10(3):643–657, 2000.