Scientific Research Publishing

Open Journal of Optimization, 2021, 10, 47-60
https://www.scirp.org/journal/ojop
ISSN Online: 2325-7091
ISSN Print: 2325-7105

Experimental Study of Methods of Scenario Lattice Construction for Stochastic Dual Dynamic Programming

Dmitry Golembiovsky[1], Anton Pavlov[1], Smetanin Daniil[2]

[1]Moscow State University, Moscow, Russia
[2]New Economic School, Moscow, Russia
Email: golemb@cs.msu.ru, anton980307@mail.ru, smetanindaniil97@mail.ru

How to cite this paper: Golembiovsky, D., Pavlov, A. and Daniil, S. (2021) Experimental Study of Methods of Scenario Lattice Construction for Stochastic Dual Dynamic Programming. *Open Journal of Optimization*, **10**, 47-60.
https://doi.org/10.4236/ojop.2021.102004

Received: March 28, 2021
Accepted: June 25, 2021
Published: June 28, 2021

Abstract

The stochastic dual dynamic programming (SDDP) algorithm is becoming increasingly used. In this paper we present analysis of different methods of lattice construction for SDDP exemplifying a realistic variant of the newsvendor problem, incorporating storage of production. We model several days of work and compare the profits realized using different methods of the lattice construction and the corresponding computer time spent in lattice construction. Our case differs from the known one because we consider not only a multidimensional but also a multistage case with stage dependence. We construct scenario lattice for different Markov processes which play a crucial role in stochastic modeling. The novelty of our work is comparing different methods of scenario lattice construction. We considered a realistic variant of the newsvendor problem. The results presented in this article show that the Voronoi method slightly outperforms others, but the k-means method is much faster overall.

Keywords

Stochastic Dual Dynamic Programming, Newsvendor Problem, Markov Process

1. Introduction

This article discusses one of the most powerful modern algorithms, which can solve problems of stochastic optimization—the stochastic dual dynamic programming algorithm first described in [1]. There are several ways to implement the algorithm; an exhaustive survey can be found in [2]. We will discuss the most commonly used version—SDDP algorithm with scenario lattice. The main

goal of our study is the testing of methods of a lattice construction. We considered "The problem of production, sales and product storage", which is an extension of the newsvendor problem [3], to compare the scenario lattice construction methods.

There are many articles about scenario generation methods. The approach of [4] is based on variance reduction techniques. There are works with overviews of different methods [5] [6] [7]. Multiple scenario lattice construction methods exist, but we will only consider 3 of them: k-means, competitive learning and Voronoi cell sampling. In this study, we use the scenario generation methods for stochastic variables that follow a Markovian process. We are focusing on comparing some different techniques. There were some works with a similar goal, for example [8]. However, in this work, we go further and make an empirical analysis in the stage-dependent case. We perform modeling for several days, and each next day is related to the previous day (unlike [8], where stage-independent processes are considered).

The rest of the paper is organized as follows: In Section 2, we are making an introduction to the theory, then in Sections 3 and 4 describing the methods of lattice construction we are using, and then we discuss numerical experiments in Section 5, and after this summarize all the results in Section 6.

2. Problem Formulation

The original problem of stochastic programming in general terms looks like the following [8]:

$$\min_{x\in X} C(x) \tag{1.1}$$

$$C(x) = \int c(x,z)\mathrm{d}F(z) \tag{1.2}$$

where x is a controlling variable with definition area $X \subseteq \mathbb{R}^n$; z is a vector vector of realizations of random variable Z; $C(x)$ is a cost function.

To resolve this problem, we need to move from integral, whose computation is too hard, to sum, so our problem will have the discrete form:

$$\min_{x\in X} \hat{C}(x) \tag{1.3}$$

$$\hat{C}(x) = \sum_{\hat{z}\in\hat{F}} \hat{p}(z)c(x,\hat{z}) \tag{1.4}$$

In literature, this transition is known as sample average approximation (SAA), and it is formally described, for example, in [9].

There are many ways to implement transition from (1.1)-(1.2) to (1.3)-(1.4), for example Monte Carlo [10] [11] or quasi-Monte Carlo methods [12]. We will use another method based on the direct reduction of the approximation error of $\hat{C}(x)$ to $C(x)$, which was described in [12]. We will briefly describe this method below.

3. Methods and Algorithms of Probability Distribution Discretization

Our approach is based on established work [4]. We will briefly discuss it here.

Our task is to approximate the distribution F with discrete one $\hat{F}$. To do so, we generate realizations of F. The Monte-Carlo method provides the approximation cost function convergence to the real cost function with probability of 1, but we want to reduce the variance of the estimate to speed up the convergence of the error bounds. To do so, we want to address the approximation error directly. Let's define an error in SAA as absolute difference between the cost functions:

$$e\left(C,\hat{C}\right)=\left|\max_x\left(C(x)-C\left(\arg\min_x \hat{C}(x)\right)\right)\right| \tag{1.5}$$

Since it is nearly impossible to compute $e\left(C,\hat{C}\right)$ with (1.5), it is usual to work with the upper bound of the error. To estimate the upper bound, firstly, we need to introduce some additional notation. Denote as $L_r\left(c(x)\right)$ Lipschitz constant of $c(x,z)$ function of r order:

$$\begin{aligned} L_r\left(c(x,z)\right)=\inf\Big\{L:&\left|c(x,z_1)-c(x,z_2)\right| \\ &\le L\left|z_1-z_2\right|\max\left(1,\left|z_1\right|^{r-1},\left|z_2\right|^{r-1}\right)\ \forall z_1,z_2\in Z\Big\}\le \bar{L}_r \end{aligned} \tag{1.6}$$

where $\bar{L}_r$ is the upper bound. Further, d_r is Wasserstein distance between F and $\hat{F}$ distributions:

$$d_r\left(F,\hat{F}\right)=\left(\min_{g\in M\left(F,\hat{F}\right)}\int\left\|z-\hat{z}\right\|^r \mathrm{d}g\left(z,\hat{z}\right)\right)^{\frac{1}{r}}, \tag{1.7}$$

where the minimum is taken over the entire space of distribution functions $M\left(F,\hat{F}\right)$, marginal distributions F and $\hat{F}$ are Lipschitz, which means they satisfy (1.6). From [12] we know that:

$$e\left(C,\hat{C}\right)\le 2\bar{L}_r d_r\left(F,\hat{F}\right), \tag{1.8}$$

so as far as $\bar{L}_r$ is a constant for given r and $c(x,z)$, it is clear that with the reduction of $d_r\left(F,\hat{F}\right)$, the approximation error will also reduce.

It is impractical to find $d_r\left(F,\hat{F}\right)$ using (1.7) because of the integral, so we will use distance between two discrete distributions:

$$d_r\left(F,\hat{F}\right)=\left(\min_{y_{i,j}\in[0,1]}\left(\sum_{i=1}^N\sum_{j=1}^M y_{i,j}\left\|z_i-\hat{z}_j\right\|^r \mid \sum_{j=1}^M y_{i,j}=p_i,\sum_{i=1}^N y_{i,j}=q_j\right)\right)^{\frac{1}{r}}, \tag{1.9}$$

where p_i,q_j are the probabilities of the corresponding discrete distribution values, and $y_{i,j}=p_iq_j$. Thus, according to (1.8), we need to find the discrete distribution $\hat{F}$, which is the best approximation for F, reducing SAA error.

Since we consider continuous distributions, we need to generate a sample of size N, so (1.9) will have the following form:

$$\min_{\hat{z}_1,\cdots,\hat{z}_M}\left(\left(\min_{y_{i,j}\in\{0,1\},q_j\ge 0}\left(\sum_{i=1}^N\sum_{j=1}^M y_{i,j}\left\|z_i-\hat{z}_j\right\|^r \mid \sum_{j=1}^M y_{i,j}=1,\sum_{i=1}^N y_{i,j}=q_jN\right)\right)^{\frac{1}{r}}\right), \tag{1.10}$$

where $q_j, j=1,\cdots,M$ are probabilities of discrete distribution values; $y_{i,j}=1$, if element z_i of the original sample attributes to element $\hat{z}_j$ of the new dis-

tribution, otherwise $y_{i,j} = 0$; M is the number of values of the new distribution, $M < N$.

Now, we will briefly describe algorithms, which we used to solve (1.10) and get a new approximate probability distribution. More comprehensively, they are discussed in [8].

k-means algorithm:

1) Randomly choose M elements $\hat{z}_1^*, \cdots, \hat{z}_M^*$ from the source sample $z_1, \cdots, z_N$.

2) Attribute element $i, i = 1, \cdots, N$ of sample $z_1, \cdots, z_N$ to element j of sample $\hat{z}_1^*, \cdots, \hat{z}_M^*$, if j is the solution of $\min_j \left\| z_i - \hat{z}_j^* \right\|$.

3) Recalculate cluster centers $\hat{z}_1^*, \cdots, \hat{z}_M^*$: $\hat{z}_j^* = \frac{1}{S_j} \sum_{k=1}^{s_j} z_k$, where s_i is the number of elements of the sample $z_1, \cdots, z_N$, attributed to cluster j on stage 2.

4) If the stopping condition is met, then stop, else return to step 2.

Note that we are using k-means modification from [13].

Competitive learning:

The sample $z_1, \cdots, z_N$ is obtained from the distribution F and initial approximations $\left(\hat{z}_j^0\right)_{j=1}^M$ are chosen. Then the following algorithm is fulfilled for $j = 1, \cdots, M; i = 1, \cdots, N$:

$$\hat{z}_j^n = \begin{cases} \hat{z}_j^{n-1} + \alpha_n \left(z_n - \hat{z}_j^{n-1} \right), & \text{if } j = \arg\min_k \left(\left\| z_n - \hat{z}_k^{n-1} \right\|^2 \right) \\ \hat{z}_j^{n-1}, & \text{else} \end{cases} \tag{1.11}$$

α_n is the step size, $j = 1, \cdots, M; n = 1, \cdots, N$, values $\hat{z}_j^N$ give the wanted discrete distribution.

Voronoi cells sampling:

The sample $z_1, \cdots, z_N$ is obtained from the distribution F and initial approximations $\left(\hat{z}_j^0\right)_{j=1}^M$ are chosen. Then the following algorithm is fulfilled for $j = 1, \cdots, M; i = 1, \cdots, N$:

$$\hat{z}_j^n = \begin{cases} \hat{z}_j^{n-1} + \alpha_n \left(z_n - \hat{z}_j^{n-1} \right), & \text{if } j = \arg\min_k \left(\left\| z_n - \hat{z}_k^{n-1} \right\|^2 \right) \\ \hat{z}_j^{n-1}, & \text{else} \end{cases} \tag{1.12}$$

Further, put $\hat{z}_j'^0 = \hat{z}_j^0$ and:

$$\hat{z}_j'^n = \begin{cases} z_n, & \text{if } j = \arg\min_k \left(\left\| z_n - \hat{z}_k^{n-1} \right\|^2 \right) \\ \hat{z}_j'^{n-1}, & \text{else} \end{cases} \tag{1.13}$$

4. Scenario Lattice Construction Algorithm

Now consider the scenario lattice construction algorithm that was used.

1) First of all, the grid parameters are selected: the number of stages, the number of nodes at each stage, the number of scenarios generated from each node.

2) Then, from each vertex of stage, t, we generate M scenarios according to the

relevant random processes.

3) From the $M \times N$ scenarios, where N is the number of nodes in the t stage, N is selected using one of the methods described in Section 3.

4) The probabilities of transition from the vertex i of stage t to the vertex j of stage $t+1$ are calculated as the ratio of the number of scenarios generated from the vertex i and related to the vertex j of stage $t+1$ to the total number of scenarios generated from the vertex i.

5) If $t+1$ is not equal to T, where T is the number of stages, we pass to point 2; otherwise, the grid is constructed.

The lattice obtained with this algorithm can be illustrated as follows (**Figure 1**):

In **Figure 1**, each vertex represents a vector of a given dimension, corresponding to random variables. The probability of transition from the node i of stage t to the node j of stage $t+1$ is indicated by $p_{i,j}^{t}$. Some probabilities can be zero, and since we are using clustering algorithms to group scenarios, our lattice can be illustrated by **Figure 2**. The vertex of step 1 corresponds to the initial values, which are selected before the beginning of the grid construction algorithm. We are using standard method [1] to choose scenarios during the forward step of the SDDP algorithm.

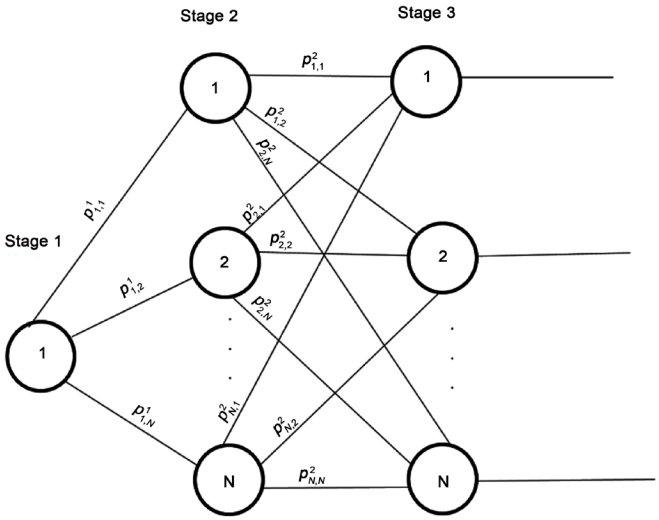

Figure 1. Scenario lattice.

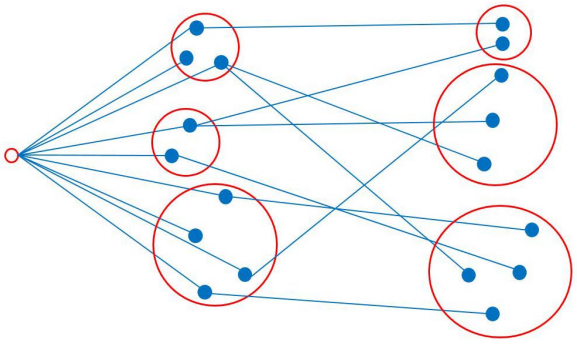

Figure 2. Scenario lattice construction.

5. The Case Problem

To compare k-means, competitive learning, and Voronoi cell sampling algorithms, we will use them to solve "The problem of commodities production, storage and selling" that can be formulated as follows.

Let T be the time interval at which we consider the problem, k is the number of type of goods, p^i is the cost of single item production of type i, v^i is the maximum number of items that can be produced in one day, x_t^i is the number of goods of type i that were produced on day t, s^i is the selling price for goods of type i, δ_t^i is the demand for type of goods i on day t, r_t^i – number of undelivered goods of type i on day t, c^i is the cost of storage of goods of type i.

The goal is to find a strategy (amount of produced goods) that leads to maximum income.

Income in day t: $\sum_{i=1}^{k}\left(-p^i x_t^i + s^i w_t^i - c^i r_t^i\right)$

Amount of sold goods in day t: $w_t^i = \min\left(\delta_t^i, x_t^i + r_{t-1}^i\right), i = 1,\cdots,k$

Amount of goods in storage at the end of day t: $r_t^i = r_{t-1}^i + x_t^i - w_t^i, i = 1,\cdots,k$

CVaR is used as a risk measure; the optimization criterion is the weighted sum of CVaR and profit expectation: $(1-\lambda)\mathbb{E}[P] + \lambda\text{CVaR}$.

Formally dynamic programming equations look like the following:

for $t = T,\cdots,2$

$$\mathcal{Q}_t\left(r_{t-1},\delta_t\right) = \min_{\substack{x_t^i \le v^i \\ w_t^i \le \delta_t^i \\ w_t^i \le x_t^i + r_{t-1}^i \\ r_t^i = r_{t-1}^i + x_t^i - w_t^i \\ x_t^i \ge 0, u_t \ge 0}} \left\{\sum_{i=1}^{k}\left(-p^i x_t^i + s^i w_t^i - c^i r_t^i\right) + \lambda_{t+1}u_t + Q_{t+1}\left(r_t,u_t,\delta_t\right)\right\}$$

where

$Q_t\left(r_{t-1},u_t-1,\delta_{t-1}\right) = \mathbb{E}\left\{(1-\lambda_t)\mathcal{Q}_t\left(r_{t-1},\delta_t\right) + \lambda_t\alpha_t^{-1}\left[\mathcal{Q}_t\left(r_{t-1},\delta_t\right) - u_{t-1}\right]_+ \mid \delta_{t-1}\right\}$,

($Q_{T+1}(.,.,.) \equiv 0$ by definition), $r_t = \left(r_t^1,\cdots,r_t^k\right)$ и $\delta_t = \left(\delta_t^1,\cdots,\delta_t^k\right)$.

for $t = 1$

$$\min_{\substack{x_1^i \le v^i \\ w_1^i \le \delta_1^i \\ w_1^i \le x_1^i + r_0^i \\ r_1^i = r_0^i + x_1^i - w_1^i \\ x_1^i \ge 0, u_1 \ge 0}} \left\{\sum_{i=1}^{k}\left(-p^i x_1^i + s^i w_1^i - c^i r_1^i\right) + \lambda_2 u_t + Q_2\left(r_1,u_1,\delta_1\right)\right\}$$

6. Numerical Experiments

For numerical experiments, we considered four different stochastic processes for the demand for goods.

1) Autoregressive model (AR):

$$X_t = c + \sum_{i=1}^{p} a_i X_{t-i} + \varepsilon_t,$$

where c is a constant, α_i are the model parameters, $\varepsilon_t \sim N\left(0,\sigma^2\right)$

2) Autoregressive moving-average model (ARMA):

$$X_t = c + \sum_{i=1}^{p} \alpha_i X_{t-i} + \varepsilon_t + \sum_{j=1}^{q} \beta_j \varepsilon_{t-j},$$

where c is a constant, α_i, β_j are the model parameters, $\varepsilon_t \sim N\left(0, \sigma^2\right)$

3) Geometric Brownian motion (GBM):

$$\mathrm{d}S_t = \mu S_t + \sigma S_t \mathrm{d}W_t,$$

where μ, σ are process parameters, W_t is the Wiener process.

4) Stage independent normal distribution (SIND):

$$X_t = N\left(\mu, \sigma^2\right)$$

We are using the processes with chosen parameters, so in our case, the processes look as follows:

1) GBM: $\mathrm{d}X_t = 5X_t \mathrm{d}t + 0.1X_t \mathrm{d}W_t$;

2) AR(1): $X_t = c + 0.9X_{t-1} + \varepsilon_t, \varepsilon_t \sim N\left(0, \sigma^2\right), \sigma = 1$;

3) ARMA(1,1): $X_t = c + \varepsilon_t + 0.9X_{t-1} + 0.15\varepsilon_{t-1}, \varepsilon_t \sim N\left(0, \sigma^2\right), \sigma = 1$;

4) SIND: $X_t \sim N\left(\mu, \sigma^2\right), \mu = 10, \sigma = 5$.

The experiments were organized as follows. First of all, for every combination of process (AR, ARMA, GBM, SIND) and scenario grid construction algorithm, we simulated the stochastic process several times. We then checked if the results came from the normal distribution using the Shapiro-Wilk test and quantile-quantile (Q-Q) plot, then compared results using the one-sided t-test. (Q-Q) plots show the relationship between observed data and theoretical quantiles. It is necessary to check our results for normality because we are using the t-test to compare average profits obtained using different lattice construction methods. All the parameters of our case problem are in Table 1. We have chosen these parameters so as to simulate the real-world situation.

For the SDDP algorithm, we used the following optimization criterion:

$$0.5\mathbb{E}[P] + 0.5\mathrm{CVaR}$$

Our lattice construction parameters:

The number of nodes at each stage is 10; number of scenarios generated from each node during lattice construction is 100.

To perform the experiments, we were using a computer with an Intel Core i5-6300HQ processor running at 2.30 GHz.

We used Python 3.7.3 to generate scenario lattice and Julia 1.3.1 to run the SDDP algorithm; we implemented the SDDP by ourselves. To understand SDDP realization, please refer to [14].

The full list and versions of used packages are in Table 2 for Python and in Table 3 for Julia.

Table 1. Parameters of experiments.

Number of commodities	1	2	3	4	5	6	7	8	9
Production costs p_i	150	80	30	17	40	10	30	10	20
Selling price s_i	200	100	40	22	70	13	60	12	27
Storage cost c_i	30	3	4	2	10	2	7	1.5	4
Maximum production volume v_i	10	20	40	70	100	120	160	200	300

Table 2. Python packages.

Numpy	1.16.2
Pandas	0.24.2
Scipy	1.2.1
Sklearn	0.22.1

Table 3. Julia packages.

JuMP	0.21.1
Clp	0.7.1
Distributions	0.22.5
CSV	0.5.26

6.1. Numerical Experiments

First of all, it is necessary to compare lattice construction time for every process and lattice construction algorithm because work time plays a crucial role in computing.

As we can see (**Figure 3**), k-means works much faster than the Voronoi and Competitive learning methods, whose results are almost the same.

6.2. AR Demand Results

As we can see from the tables and plots, our results came from a normal distribution (**Figures 4-6** and **Table 4**), and mean profit from scenarios generated by the Voronoi method is higher than the profit of scenarios generated by k-means and Competitive learning (**Table 4** and **Table 5**). (Element *i,j* of **Table 5** means p-value of t-test, which compares method in row *i* and the method in column *j*).

6.3. ARMA Demand Results

In this case, our results are also normal (see **Figures 7-9** and **Table 6**), but there is no difference between k-means and Voronoi, while Competitive learning results are terrible (**Table 6** and **Table 7**).

6.4. GBM Demand Results

For GBM process, our results are slightly different from the normal distribution but not so much (**Figures 10-12** and **Table 8**), and all three methods show almost the same result (**Table 8** and **Table 9**).

6.5. Stage-Independent Normal Distribution Demand Results

For the Competitive learning method, our results slightly differ from the normal distribution but not so much (**Figures 13-15** and **Table 10**). The k-means and Voronoi methods show almost identical results, while Competitive learning loses pretty badly (**Table 10** and **Table 11**).

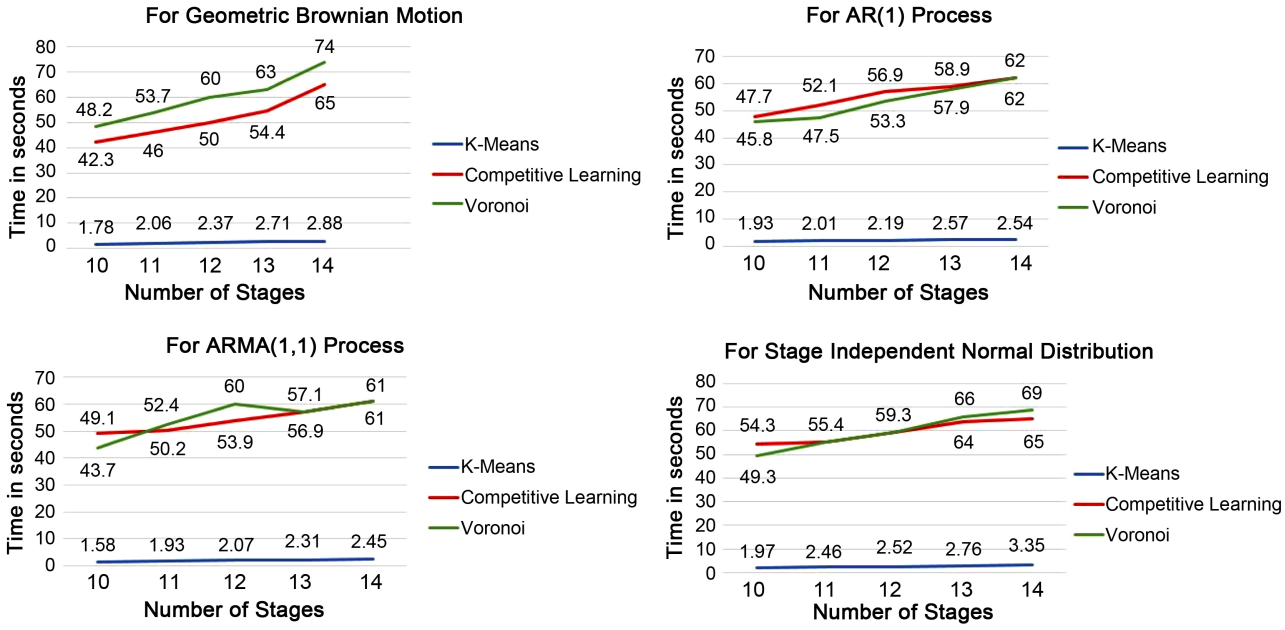

Figure 3. Lattice construction time.

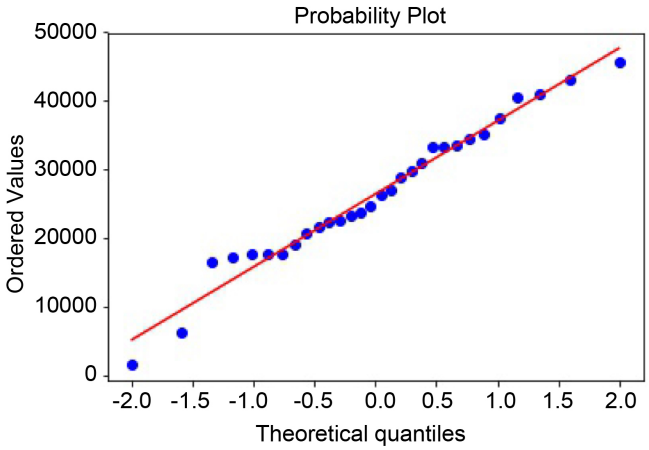

Figure 4. K-means Q-Q plot.

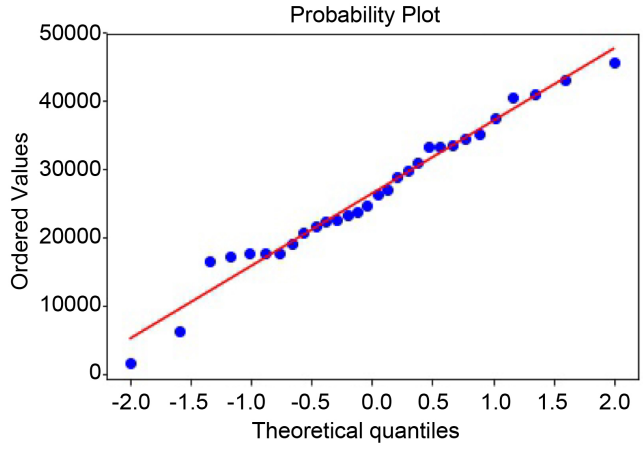

Figure 5. Competitive learning Q-Q plot.

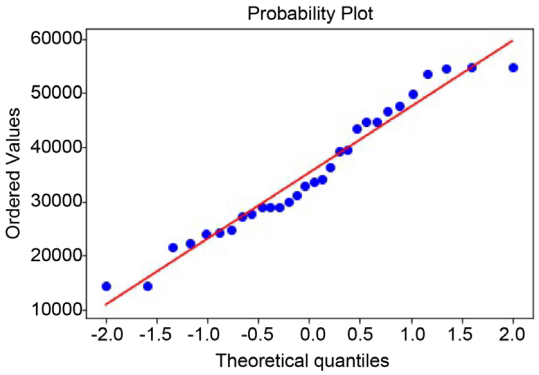

Figure 6. Voronoi Q-Q plot.

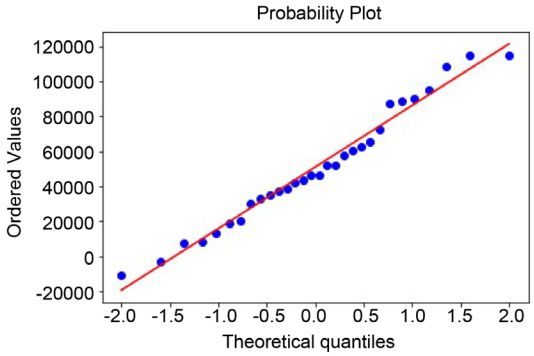

Figure 7. K-means Q-Q plot.

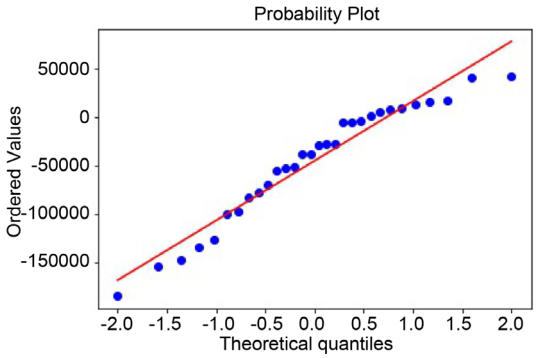

Figure 8. Competitive learning Q-Q plot.

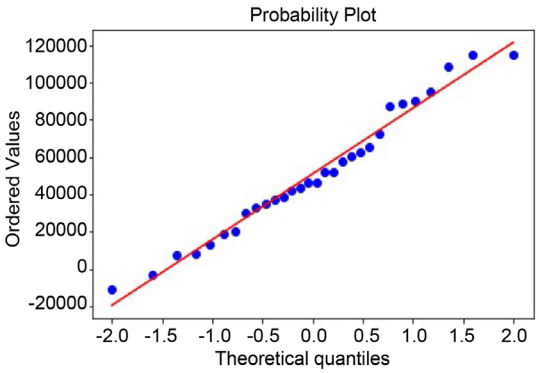

Figure 9. Voronoi Q-Q plot.

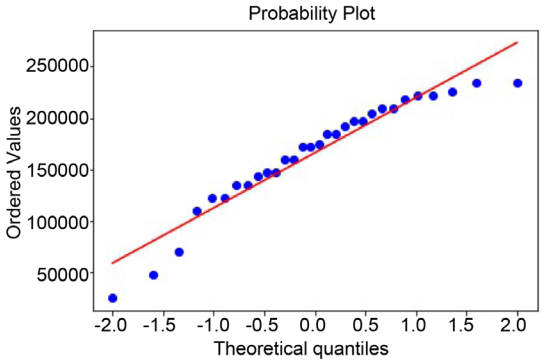

Figure 10. K-means Q-Q plot.

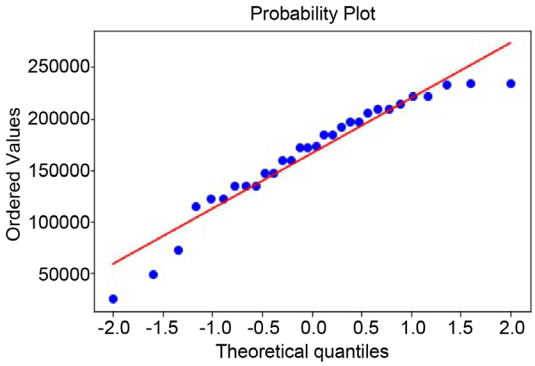

Figure 11. Competitive learning Q-Q plot.

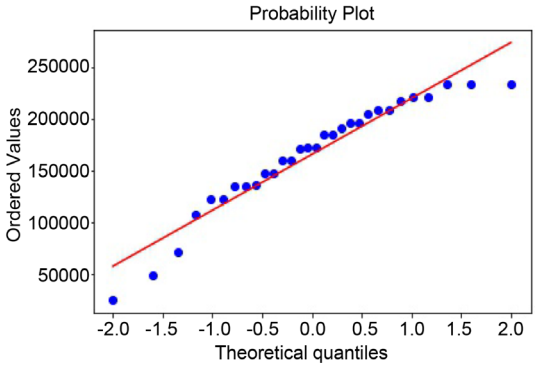

Figure 12. Voronoi Q-Q plot.

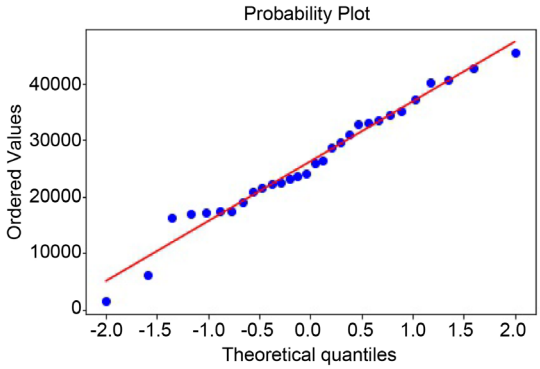

Figure 13. K-means Q-Q plot.

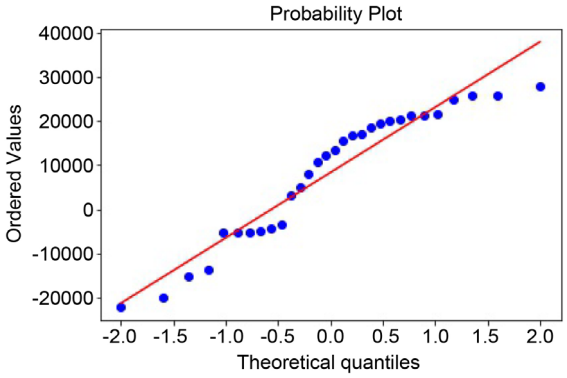

Figure 14. Competitive learning Q-Q plot.

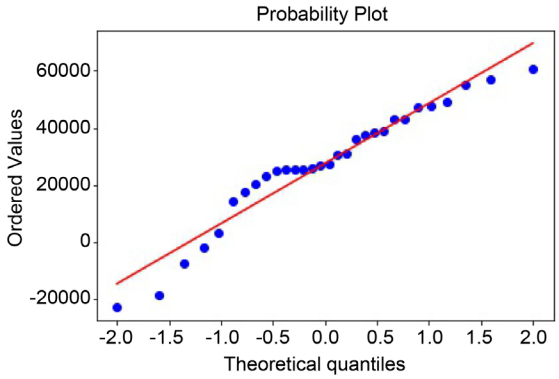

Figure 15. Voronoi Q-Q plot.

Table 4. AR statistics.

	K-means	Competitive learning	Voronoi
Mean	26,390.746	26,390.623	35,290.404
SD	10,196.322	10,196.386	11,703.831
Shapiro-Wilk p-value	0.688	0.688	0.182

Table 5. AR t-test p-values.

	K-means	Competitive learning	Voronoi
K-means	-	0.499	0.001
Competitive learning	0.499	-	0.001
Voronoi	0.001	0.001	-

Table 6. ARMA statistics.

	K-means	Competitive learning	Voronoi
Mean	50,931.03	−44,927.187	50,900.551
SD	33,633.708	59,910.994	33,635.331
Shapiro-Wilk p-value	0.548	0.091	0.542

Table 7. ARMA t-test p-values.

	K-means	Competitive learning	Voronoi
K-means	-	8.17e−10	0.498
Competitive learning	8.17e−10	-	8.24e−10
Voronoi	0.498	8.24e−10	-

Table 8. GBM statistics.

	K-means	Competitive learning	Voronoi
Mean	166,051.19	166,163.13	165,994.99
SD	52,557.44	52,572.10	52,972.71
Shapiro-Wilk p-value	0.029	0.036	0.037

Table 9. GBM t-test p-values.

	K-means	Competitive learning	Voronoi
K-means	-	0.496	0.498
Competitive learning	0.496	-	0.495
Voronoi	0.498	0.495	-

Table 10. Stage-independent statistics.

	K-means	Competitive learning	Voronoi
Mean	26,243.39	8323.40	27,525.61
SD	10,142.48	14,519.86	20,496.62
Shapiro-Wilk p-value	0.713	0.019	0.099

Table 11. Stage-independent t-test p-values.

	K-means	Competitive learning	Voronoi
K-means	-	7.08e−07	0.382
Competitive learning	7.08e−07	-	6.85e−05
Voronoi	0.382	6.85e−05	-

7. Conclusions

As we can see in the numerical results section, in our experimental problem, the Voronoi method slightly outperforms others, but the k-means method is much faster.

1) With the AR process, the Voronoi method performs better than k-means and Competitive learning, which both show almost the same result.

2) With the ARMA process, the Voronoi method is comparable to k-means, and Competitive learning underperforms pretty hard.

3) With GBM process, results of the methods are close to each other.

4) Scenario lattice calculation time was the same for the Competitive learning and Voronoi method but much lower for the k-means.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Pereira, M.V.F. and Pinto, L.M.V.G. (1991) Multi-Stage Stochastic Optimization Applied to Energy Planning. *Mathematical Programming*, **52**, 359–375. https://doi.org/10.1007/BF01582895

[2] Dentcheva, D. and Ruszczynski, A. (2009) Lectures on Stochastic Programming. Modeling and Theory. *Society for Industrial Mathematics*, **9**, 271-279.

[3] Birge, J.R. and Louveaux, F. (2011) Introduction to Stochastic Programming. Springer, NewYork. https://doi.org/10.1007/978-1-4614-0237-4

[4] Higle, J. L. (1998) Variance Reduction and Objective Function Evaluation in Stochastic Linear Programs. *INFORMS Journal on Computing*, **10**, 121-260. https://doi.org/10.1287/ijoc.10.2.236

[5] Kaut, M. and Wallace, S. (2007) Evaluation of Scenario-Generation Methods for Stochastic Programming. *Pacific Journal of Optimization*, **3**, 257-271.

[6] Roemich, W. and Heitsch, H. (2009) Scenario Tree Modelling for Multi-Stage Stochastic Programs. *Mathematical Programming*, **118**, 371-406. https://doi.org/10.1007/s10107-007-0197-2

[7] Vazsonyi, M. (2006) Overview of Scenario Tree Generation Methods, Applied in Financial and Economic Decision Making. *Periodica Polytechnica Social and Management Sciences*, **14**, 29-37. https://doi.org/10.3311/pp.so.2006-1.04

[8] Löhndorf, N. (2016) An Empirical Analysis of Scenario Generation Methods for Stochastic Optimization. *European Journal of Operational Research*, **255**, 121-132. https://doi.org/10.1016/j.ejor.2016.05.021

[9] Shapiro, A. (2013) Sample Average Approximation. In: Gass, S.I. and Fu, M.C., Eds., *Encyclopedia of Operations Research and Management Science*, Springer, Boston, MA, 1350-1355. https://doi.org/10.1007/978-1-4419-1153-7_1154

[10] Shapiro, A. (2003) Monte Carlo Sampling Methods. *Handbooks in Operations Research and Management Science*, **10**, 353-425. https://doi.org/10.1016/S0927-0507(03)10006-0

[11] Glasserman, P. (2004) Monte Carlo Methods in Financial Engineering. Springer, NewYork. https://doi.org/10.1007/978-0-387-21617-1

[12] Pflug, G.C. (2001) Scenario Tree Generation for Multiperiod Financial Optimization by Optimal Discretization. *Mathematical Programming*, **89**, 251-271. https://doi.org/10.1007/PL00011398

[13] Arthur, D. and Vassilvitskii, S. (2007) K-means++: The Advantages of Careful Seeding. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, *SODA* 2007, New Orleans, Louisiana, USA, 7-9 January 2007, 1027-1035.

[14] Dowson, O. and Kapelevich, L. (2020) SDDP.jl: A Julia Package for Stochastic Dual Dynamic Programming. *INFORMS Journal on Computing*, **33**, 1-418. https://doi.org/10.1287/ijoc.2020.0987