Proceedings of the 2021 IISE Annual Conference
A. Ghate, K. Krishnaiyer, K. Paynabar, eds.

Exploring the Effect of Clustering Algorithms on Sample Average Approximation

Abstract ID: 972645

Daniel Jacobson
Grado Department of Industrial and Systems Engineering,
Virginia Tech, Falls Church, VA 22043

Menna Hassan
School of Electrical and Computer Engineering, Purdue University, Lafayette, IN 47907

Zhijie Sasha Dong
Ingram School of Engineering, Texas State University, San Marcos, TX 78666

Abstract

Stochastic Programming (SP) is used in disaster management, supply chain design, and other complex problems. Many of the real-world problems that SP is applied to produce large-size models. It is important but challenging that they are optimized quickly and efficiently. Existing optimization algorithms are limited in capability of solving these larger problems. Sample Average Approximation (SAA) method is a common approach for solving large scale SP problems by using the Monte Carlo simulation. This paper focuses on applying clustering algorithms to the data before the random sample is selected for the SAA algorithm. Once clustered, a sample is randomly selected from each of the clusters instead of from the entire dataset. This project looks to analyze five clustering techniques compared to each other and compared to the original SAA algorithm in order to see if clustering improves both the speed and the optimal solution of the SAA method for solving stochastic optimization problems.

Keywords
Stochastic Programing; Sample Average Approximation; Clustering, Machine Learning, Heuristic Algorithm

1. Introduction
Stochastic Programming (SP) [1] is defined as modeling optimization problems in which a portion of the dataset being inputted into the objective function or constraints is uncertain. SP is commonly used in disaster management, supply chain design, and other complex problems. Many of the real-world problems that SP is applied to, produce large models, and it is important that they are optimized quickly and efficiently. Applications include disaster supply management [2], supply chain design [3], and healthcare [4], all of which require highly efficient and optimal solutions to extremely complex problems. Although the data for this research was related to disaster supply management, this work focuses more on the optimization of the SAA algorithm than its application.

Existing optimization algorithms are limited in the capability of solving these large problems. Sample Average Approximation (SAA) method is a common approach for solving large scale SP problems by using the Monte Carlo simulation. SAA approximates the SP objective function by a sample average estimation derived from a random sample. The resulting SAA problem is solved deterministically. The process is repeated with different random samples to obtain potential solutions.

This paper proposes a method to create a machine learning based computational framework. Many research studies focus on applying machine learning algorithms to improve SP optimization. For example, Cotter's work (2013) improves stochastic algorithm by applying two common machine learning algorithms: binary classification using kernelized linear classifiers and Principle Component Analysis [5]. In another study by Verweij et al. (2013), the SAA algorithm was applied to stochastic routing problems and found to have a computational complexity that grows

linearly with the number of sample scenarios [6]. The studies concluded that SAA can become computationally expensive for more complex and larger datasets. In the case of this study, disaster management data was used, which is a large multidimensional dataset. The utilization of machine learning ideas to the optimization algorithms such as heuristic algorithms for SP problems, is not as common. This project focused on applying clustering algorithms to the data before the random sample is selected. Once clustered, the random sample is randomly selected from each of the clusters instead of from the entire dataset.

This paper is outlined as follows: Section 2 focuses on literature relating to the model and integration of machine learning into the SAA algorithm. Section 3 details the model itself and the data being used as input. Section 4 overviews the various clustering methods and some of their advantages and disadvantages. Lastly, Section 5 and 6 explain the results and conclusions of this work.

2. Literature Review

2.1 Model

Heuristic algorithms are those that focus more on reaching an approximate solution for the sake of speed, sometimes sacrificing the exact solution. Heuristic algorithms are commonly used to solve stochastic programming problems. The use of SAA algorithms, as one of efficient heuristic algorithms, to solve stochastic programming problems is quite common. Kleywegt et al. (2002) apply the SAA algorithm to solve stochastic discrete optimization problems. The study analyzes convergence rates and computational complexity of the SAA algorithm [7].

It is important that solutions should be found to reduce the computational expense of running these algorithms without sacrificing the overall performance of the objective function. Many scenario reduction methods [8] are applied to the SAA algorithm to enhance the computational efficiency of this algorithm. Pasupathy and Song (2021) present an adaptive sequential SAA algorithms to solve large-scale two-stage stochastic linear programs [9]. Henrion et al. (2009) develop a scenario reduction method based on discrepancy distances to derive the upper and lower bounds, and some explicit solutions for optimal scenario reduction problems [10].

2.2 Integration of Machine Learning Algorithms

Studies that have tried to resolve this issue utilize clustering algorithms to group similar scenarios together, and from this, generate samples. This idea can be seen in studies by Crainic et al. (2014)[11] and Emelogu et al. (2016)[12]. This technique was seen to generate samples more efficiently than the original SAA algorithm. It is common for optimization algorithms to be used to improve machine learning techniques, but not the other way around. Although the SAA algorithm achieve good performance, the computational expense makes it difficult and inefficient to use on much larger problems that can be seen in disaster management and other real world situations. This paper explores machine learning based clustering algorithms to more efficiently generate samples for the SAA algorithm.

3. Stochastic Programming Model and Data

3.1 Data

The data for this study are extracted from a facility location problem which allocated certain products to given facilities. The products had a demand and volume associated with them, as well as a procurement, transportation, holding, and shortage cost. The facilities had given storage capacities, distances to locations, and fixed costs associated with them. The decision variables included whether to open a facility. Determining how much product should be held at each facility as well as the amount of product to deliver from an open facility to a designated location were also decision variables. These variables correlated to the fixed, procurement, and transportation costs of operating an open facility. Holding costs are also relevant for products that are being held in a location, and shortage costs are relevant when a product runs out at a given facility. The ultimate goal of this work is to minimize the cost while improving the run time of the SAA algorithm.

3.2 Model

Notations:

N Set of locations, indexed by i, j
L Set of size categories, indexed by l
A Set of item types, indexed by a
S Set of disaster scenarios, indexed by s

$fc = \sum_{i,l} CF_l \cdot x_{i,l}$ Fixed costs for operating facilities

$pc = \sum_{a,i} CP_a \cdot y_{a,i}$ Procurement costs for all products across all facilities

$tc_s = \sum_{a,i,j} CT_a \cdot H_{i,j} \cdot q_{a,i,j,s}$ Transportation costs for all products in scenario *s*

$hc_s = \sum_{a,i,j} CH_a \cdot z_{a,i,s}$ Holding costs for surplus quantities in scenario *s*

$wc_s = \sum_{a,j} G_a \cdot w_{a,j,s}$ Penalty costs for shortage quantities in scenarios *s*

$D_{a,j,s}$ Demand for products (type *a*, location *j*, disaster scenario *s*)
p_s Probability of scenario *s* occurring
$H_{i,j}$ Distance from facility *i* to location *j*
U_l Storage capacity of facility in category *l*
CF_l Fixed cost of facility in category *l*
V_a Volume of type *a* products
CP_a Unit procurement price of type *a* products
CT_a Unit transportation cost of type *a* products
CH_a Unit holding cost of type *a* products
G_a Unit penalty cost for shortage of type *a* products
M A large positive number
$x_{i,l}$ 1 if open facility *i* in category *l*, otherwise 0.
$y_{a,i}$ Procurement quantity of type *a* products for facility *i*
$q_{a,i,j,s}$ Amount of type *a* products delivered from facility *i* to location *j* in scenario *s*

Formulation

The following is the mathematical model formulated for disaster facilities.

$$\min f = fc + pc + \sum_s p_s \left(tc_s + hc_s + wc_s \right) \quad (1)$$

$$\sum_a y_{a,i} \cdot V_a \le \sum_l U_l \cdot x_{i,l}, \quad \forall i \quad (2)$$

$$z_{a,i,s} = y_{a,i} - \sum_j q_{a,i,j,s} \quad (3)$$

$$w_{a,j,s} = D_{a,j,s} - \sum_i q_{a,i,j,s} \quad (4)$$

$$\sum_l x_{i,l} \le 1, \quad \forall i \quad (5)$$

$$x_{i,l} \in \{0,1\}, \quad \forall i,l \quad (6)$$

$$y_{a,i,s}, q_{a,i,j,s} \ge 0, \quad \forall a,i,j,s \quad (7)$$

The objective function (1) minimizes the overall cost (f), including fixed cost (fc), procurement costs (pc), transportation, costs (tc), holding costs (hc), and shortage costs (wc). Constraint (2) restricts that at each facility the total procurement quantities would not exceed the storage capacity, (3) calculates surplus quantity of type a products at facility i in scenario s. Line (4) calculates shortage quantity of type a products at location j in scenario s and the inequality (5) makes sure one facility will be operated at most. Lines (6) and (7) define the integrity of the variables.

4. Clustering Technologies

The SAA algorithm is a Monte Carlo simulation-based approach. Several clustering methods were used before the SAA algorithm's random sample selection to improve the speed and optimal solution of the SAA method. One sample scenario was selected from each of the produced clusters for input into the SAA algorithm.

4.1 K-means

The K-means algorithm requires a preset number of clusters k, each of which has a randomly generated centroid. K-means simply consists of two basic steps. First, every data point is assigned to its nearest centroid based on its Euclidean distance. Next, every centroid's average is re-calculated based on all of its assigned data points. These steps are recomputed until there is convergence. Some of the strengths of the K-means method include the fact that it is a very common and well-known algorithm. The simplicity of the algorithm also makes it fast and reaches convergence quickly. But one of the major draw backs is that the total number of clusters must be known before it is run. Lastly there are two ways to use the K-means solution in the original SAA algorithm. Stratified sampling where

the scenarios are strategically selected from each cluster and Random sampling where they are randomly selected from each cluster.

4.2 Mean-Shift

The Mean Shift algorithm begins with a centroid at every single data point. Then the centroids store all of their neighbors within a certain radius r. Then the mean of those points will be taken and the centroid will then be assigned to that point. Lastly the unique centroids are kept and stored. The three steps are repeated until convergence of the centroids. Unlike the K-means algorithm the mean-shift algorithm does not require a preset number of clusters. However, the algorithm does require a preset radius r. Additionally it can be very inefficient to begin at every data point.

4.3 Density-Based Spatial Clustering (DBSCAN)

The Density-Based Spatial Clustering Algorithm (DBSCAN) begins with a single data point. The algorithm then finds all of the points that are within a preset distance epsilon away. If fewer than a preset number of points (minPoints) are within epsilon distance, the point is labeled as noise. If more than minPoints are found, the algorithm creates a cluster of that point and all its neighboring points. Then it continuously adds all the neighbors within epsilon distance until no more points in the cluster have data points within epsilon. The algorithm repeats until all points have either been assigned as a cluster or visited and labeled as noise. The DBSCAN algorithm is excellent for identifying and omitting outliers in the dataset and does not require a preset number of clusters, unlike K-Means. However, this algorithm has two drawbacks. It does not perform well in high dimension because it is easy for a lot of points to be within an epsilon distance of each other. It also requires two preset parameters: minPoints and epsilon.

4.4 Expectation-Maximization using Gaussian Mixture Models (EM-GMM)

Gaussian Mixture Models (GMMs) assume the Gaussian distribution of clusters so that both the cluster's mean and standard deviation can describe the cluster (as opposed to just the mean being used to describe the cluster center as in K-means). GMMs give more flexibility than K-means since the assumption that clusters are Gaussian is less restrictive than the assumption that clusters are circular, which is the case in K-means. The EM (Expectation Maximization Algorithm) is used to optimize the parameters of the Gaussian distribution for each cluster. The E-step in the EM algorithm focuses on parameter initialization and probability estimation, the M-step then seeks to maximize the parameters using the probability estimates calculated in the E step.

4.5 Parameter Estimation

One of the major factors in the quality of clustering is the selection of the parameters. Each of the clustering algorithms required at least one parameter that needed to be preset before the clustering began. The assumption was that in order to get the best results out of the SAA algorithm, the samples should be chosen from robust, well distributed clusters. To select the parameters, histogram graphs were observed in order to view the distributions of the clusters. The two objectives were to not allow any one cluster to have too many data points (ideally under 25) and to have the majority of the clusters contain multiple data points (more than 1). Figure 1 shows the distribution of the data among the clusters for the K-means Random algorithm with parameter k = 18. The clusters appear to satisfy both of the objectives. The largest cluster only has 15 data points and only one cluster appears to have one data point. Table 1 displays the results of the parameter estimation. The DBSCAN algorithm was the hardest to estimate. The best results only produced eight clusters, one of which contained almost half of the data. The remaining algorithms all had the number of clusters between 13 and 21, with the largest cluster containing between 16 and 21 data points.

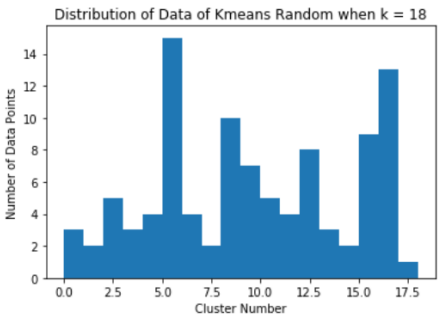

Figure 1: The distribution of data for K-Means Random when k = 18

Table 1: Summary of the parameter selection for each algorithm

Clustering Algorithm	Parameter(s)	Parameter Value(s)	Number of Clusters	Largest Cluster
Kmeans Random	k	18	18	16
Kmeans Stratified	k	13	13	18
DBSCAN	(minP, eps)	(3, 650)	8	49
Mean Shift	r	1050	21	21
EM Gauss	k	15	15	20

5 Results

The variables this project focused on for model comparison were the average optimization time and the average optimal solution of the SAA algorithm. Each algorithm found an optimal solution 50 times. The boxplots of the optimal solution for each algorithm are displayed in figure 2. The Original SAA algorithm appears to have the smallest distribution. The EM-GMM and Mean-Shift algorithms also have small distributions but have both more outliers and outliers that are farther away from the mean. The Original SAA algorithm has the smallest average optimal solution of 31.57 million as well as the smallest variance at 0.98 million. No other algorithm was able to achieve an average optimal solution below 32 million. EM-GMM was the next best algorithm in terms of both average optimal solution and standard deviation with the second smallest average of 32.45 million and the second smallest standard deviation of 2.99 million. DBSCAN and K-means stratified had the two largest averages of 33.31 and 33.99 million and two largest standard deviations of 3.36 and 3.43 million respectively.

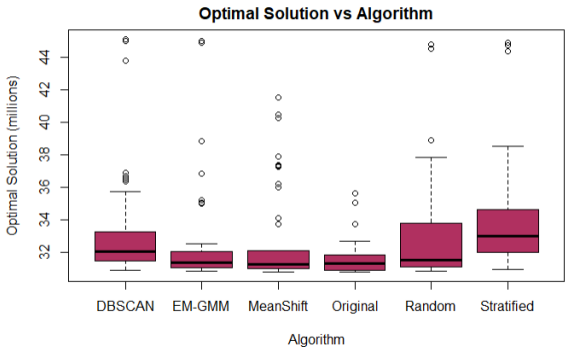

Figure 2: Boxplots of the optimal solution for each algorithm.

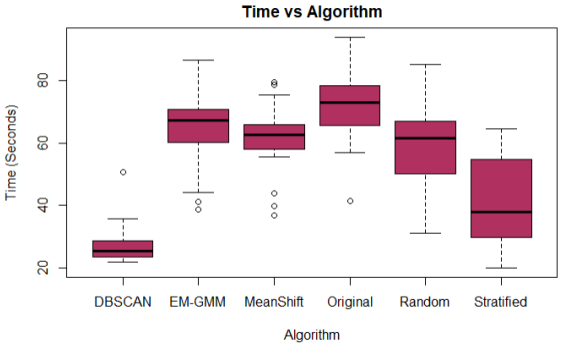

Figure 3: Boxplots of the time to solve for each algorithm.

The ultimate goal of the SAA algorithm is to reduce compute time. When comparing clustering algorithms that assist the SAA algorithm, time to solve the stochastic problem was also considered. Figure 3 shows the boxplots of the time each algorithm took to solve the problem for each of the 50 trials. DBSCAN has a significant advantage in speed compared to the other algorithms. K-means Stratified and K-means random have the largest distribution of times. The Original SAA algorithm has the slowest time. The DBSCAN algorithm proved to be the fastest averaging 26.75 seconds with the lowest standard deviation of 4.82 seconds. The K-means Stratified algorithm was second fastest averaging 40.13 seconds but with the highest standard deviation of 13.09 seconds. The Original SAA algorithm was the slowest averaging 72.28 seconds. This is an entire 7 seconds slower than the next slowest algorithm, EM-GMM. Table 2 shows the detailed statistics of the optimal solutions and run times of the original SAA algorithm and the clustering methods.

Table 2: Statistics for each algorithm.

Clustering Algorithm	Average Optimal Solution (millions)	Optimal Solution Standard Deviation (millions)	Average Time (seconds)	Time Standard Deviation (seconds)
Originial SAA	31.57	0.98	72.28	9.19
Kmeans Random	32.78	3.06	58.32	11.83
Kmeans Stratified	33.99	3.43	40.13	13.09
DBSCAN	33.31	3.36	26.75	4.82
Mean Shift	32.72	2.85	62.10	7.73
EM Gauss	32.45	2.99	65.21	9.89

Tukey's range test is used to pairwise compare all of the means and determine which, if any, are significantly different from the rest. Tukey's test was performed on both the mean optimal solutions and the mean time. The results of Tukey's test can be viewed in Tables 3 and 4. The first column shows which two algorithm means are subtracted from each other. The second column shows the difference between the means. The third and fourth columns give the lower and upper bounds of the difference in means estimation. Lastly, the fifth column gives an adjusted p-value for the likelihood the event occurs. Using $\alpha = 0.05$, there are only two occurrences in which one algorithm performed statistically significantly better than the next. The original SAA algorithm has a statistically lower average optimal solution than the DBSCAN algorithm by an average difference of 1.74 million. Likewise, the original SAA algorithm has a significantly lower average optimal solution than K-means Stratified by an average difference of 2.42 million. Tukey's range test on the average times revealed a much different story. Almost all of the algorithms

Table 3: Tukey's range test on the mean optimal solution.

$\mu_i - \mu_j$	diff	lwr	upr	p adj
EM-GMM-DBSCAN	-0.86	-2.54	0.82	0.688
MeanShift-DBSCAN	-0.59	-2.27	1.09	0.917
Original-DBSCAN	-1.74	-3.42	-0.06	0.037
Random-DBSCAN	-0.53	-2.21	1.15	0.946
Stratified-DBSCAN	0.68	-1.00	2.36	0.856
MeanShift-EM-GMM	0.27	-1.41	1.95	0.997
Original-EM-GMM	-0.88	-2.56	0.80	0.659
Random-EM-GMM	0.33	-1.35	2.01	0.993
Stratified-EM-GMM	1.54	-0.14	3.22	0.095
Original-MeanShift	-1.15	-2.84	0.53	0.362
Random-MeanShift	0.06	-1.62	1.74	1.000
Stratified-MeanShift	1.27	-0.42	2.95	0.260
Random-Original	1.21	-0.47	2.89	0.306
Stratified-Original	2.42	0.74	4.10	0.001
Stratified-Random	1.21	-0.47	2.89	0.312

Table 4: Tukey's range test on the mean time to solve.

$\mu_i - \mu_j$	diff	lwr	upr	p adj
EM-GMM-DBSCAN	38.46	32.78	44.14	0.000
MeanShift-DBSCAN	35.35	29.67	41.03	0.000
Original-DBSCAN	45.53	39.85	51.21	0.000
Random-DBSCAN	31.57	25.89	37.25	0.000
Stratified-DBSCAN	13.37	7.69	19.05	0.000
MeanShift-EM-GMM	-3.11	-8.79	2.58	0.620
Original-EM-GMM	7.07	1.39	12.75	0.006
Random-EM-GMM	-6.88	-12.57	-1.20	0.008
Stratified-EM-GMM	-25.08	-30.76	-19.40	0.000
Original-MeanShift	10.18	4.50	15.86	0.000
Random-MeanShift	-3.78	-9.46	1.90	0.399
Stratified-MeanShift	-21.98	-27.66	-16.30	0.000
Random-Original	-13.96	-19.64	-8.28	0.000
Stratified-Original	-32.15	-37.84	-26.47	0.000
Stratified-Random	-18.20	-23.88	-12.52	0.000

were statistically significant from each other. DBSCAN was statically faster than all of the other algorithms. Next, K-means Stratified, which was statically slower than DBSCAN, was statistically faster than the remaining algorithms. There was no statistical difference between K-Means Random and Mean-Shift or Mean-Shift and EM-GMM. However, K-Means Random was statistically faster than EM-GMM. Lastly, the original SAA algorithm proved to be statistically significantly slower than all of the other 5 algorithms.

6 Conclusions and Future Work

This paper focuses on applying clustering algorithms to the sample generation for the SAA algorithm to enhance its computational efficiency. We compare five clustering algorithms to examine their efficacy on optimal sample selection. Based on the computational results, all five clustering algorithms increase the speed of the SAA algorithm. However, they do not appear to improve the optimal solution value by a statistically significant amount. The DBSCAN and K-means Stratified algorithms statistically worsened the average optimal solution compared to the original SAA algorithm. The SAA algorithm is applied to large scale models and often computational efficiency is equally important as the optimality of the solutions. This study ran each algorithm 50 times and concluded that using any of the five clustering algorithms, was statistically faster than using the original SAA algorithm. Further testing is needed to be done on a larger scale to understand the full tradeoffs between speed and optimal solutions. Increasing the number of scenarios for each algorithm will reduce the variance and bring more clarity to each algorithm's true average solution and average optimization time. This will help separate the algorithms that are faster and more optimal than the original SAA algorithm.

Acknowledgements

This work has been fully supported by NSF REU supplementary funding associated with grant CISE-1948159.

References

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

[2] Klibi, W., Ichoua, S., & Martel, A. (2018). Prepositioning emergency supplies to support disaster relief: A case study using stochastic programming. *INFOR: Information Systems and Operational Research*, *56*(1), 50-81.

[3] Santoso, T., Ahmed, S., Goetschalckx, M., & Shapiro, A. (2005). A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, *167*(1), 96-115.

[4] Omar, E. R., Garaix, T., Augusto, V., & Xie, X. (2015). A stochastic optimization model for shift scheduling in emergency departments. *Health Care Management Science*, *18*(3), 289-302.

[5] Cotter, A. (2013). Stochastic Optimization for Machine Learning. *arXiv preprint arXiv:1308.3509*.

[6] Verweij, B., Ahmed, S., Kleywegt, A. J., Nemhauser, G., & Shapiro, A. (2003). The sample average approximation method applied to stochastic routing problems: a computational study. *Computational optimization and applications*, *24*(2), 289-333.

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

[8] Homem-de-Mello, T., & Bayraksan, G. (2014). Monte Carlo sampling-based methods for stochastic optimization. *Surveys in Operations Research and Management Science*, *19*(1), 56-85.

[9] Pasupathy, R., & Song, Y. (2021). Adaptive Sequential Sample Average Approximation for Solving Two-Stage Stochastic Linear Programs. *SIAM Journal on Optimization*, *31*(1), 1017-1048.

[10] Henrion, R., Küchler, C., & Römisch, W. (2009). Scenario reduction in stochastic programming with respect to discrepancy distances. *Computational Optimization and Applications*, *43*(1), 67-93.

[11] Crainic, T. G., Hewitt, M., & Rei, W. (2014). Scenario grouping in a progressive hedging-based meta-heuristic for stochastic network design. *Computers & Operations Research*, 43, 90-99.

[12] Emelogu, A., Chowdhury, S., Marufuzzaman, M., Bian, L., & Eksioglu, B. (2016). An enhanced sample average approximation method for stochastic optimization. *International Journal of Production Economics*, 182, 230-252.