Safety, Reliability, Risk, Resilience and Sustainability of Structures and Infrastructure
12th Int. Conf. on Structural Safety and Reliability, Vienna, Austria, 6–10 August 2017
Christian Bucher, Bruce R. Ellingwood, Dan M. Frangopol (Editors)
©2017 TU-Verlag Vienna, ISBN 978-3-903024-28-1

Comparison of clustering strategies for the functional quantization of random functions

Graziano Fiorillo[a], Vasileios Christou[a], Amirali Shojaeian[a],
Paolo Bocchini[a], Javier Buceta[b] and Manuel J. Miranda[c]
[a]ATLSS Engineering Research Center, Department of Civil and Environmental Engineering, Lehigh University
[b]Department of Chemical and Biomolecular Engineering, Bioengineering Program, Lehigh University
[c]Department of Engineering, Hofstra University

Abstract: The accuracy of the output quantities of stochastic problems in engineering requires the use of random input that correctly represents the entire sample space of the input quantities. For example, studies involving infrastructure reliability assessment, risk estimation, and community resilience require an accurate description of the regional hazard, which through the use of simulation-based techniques could be quantified by realizations of a two-dimensional ground motion intensity measure field. The selection of a few such realizations could be made with a technique called Functional Quantization by Infinite Dimensional Centroidal Voronoi Tesselation (FQ-IDCVT). This is a technique for the optimal selection (in the mean-square sense) of a predefined number of samples of a random function with their associated weights. Although, previous applications of FQ-IDCVT demonstrate that the technique works particularly well with different types of random fields (Gaussian / non-Gaussian, homogeneous / non-homogenenous), the computational cost often becomes an issue when multi-dimensional fields are at hand, because it scales linearly with the size of the domain and the resolution that is used. To address the issue of the escalating computational cost, in this paper the implementation of several accelerated clustering techniques is investigated. A quality check for each algorithm is performed by comparing the accuracy and efficiency of each methodology against a benchmark.

1 Introduction

In almost twenty years, the interest of researchers for uncertainty quantification and number of scientific articles about probabilistic analysis grew substantially. This increased use of stochastic models in science and engineering is mainly due to the increased computational capability of modern electronic devices. However, these technological improvements are not yet sufficient to make probabilistic models the standard of practical engineering applications and, therefore, a considerable amount of research in improving the efficiency of stochastic computational methods is still necessary.

When a probabilistic simulation-based approach is adopted, several applications in civil engineering require efficient computational methods, due to both the large scale of the problem and the high number of random experiments required to accurately represent the uncertainty associated with it. Infrastructure reliability assessment, risk estimation, and community resilience analysis performed at the regional scale are typical examples of these applications [9, 6].

In all these cases, Functional Quantization by Infinite-Dimensional Centroidal Voronoi Tesselation (FQ-IDCVT) can be used to reduce the complexity of the stochastic problem. FQ-IDCVT is a technique for the optimal selection (in the mean-square sense) of a predefined number N of samples ("quanta") of a random function and their associated weights [13]. To select this optimal set of samples, an IDCVT must be built. It has been proven that the IDCVT corresponds to N regions of the functional space where the mean-square quantization error is minimized.
In previous works, FQ-IDCVT has been built using a methodology based on Lloyd's method [10]. The drawback of this algorithm is its high computational cost, which is an issue especially for applications dealing with multi-dimensional random fields or when high-resolution is required. To address this issue, several clustering techniques are investigated to compare both the computational cost and accuracy with respect to a benchmark solution.
After this introduction, the second section describes how clustering fits in the FQ-IDCVT framework. In the third section, the clustering techniques employed in the paper are described. In the fourth and fifth section the numerical results and conclusions are presented, respectively.

2 Functional Quantization by Infinite-Dimensional Centroidal Voronoi Tessellation

Functional Quantization (FQ) is a technique for the optimal representation of a random function with a finite number of samples and their weights [11]. In particular, the FQ approach consists of approximating a generic random function F with another random function F_N, which can be fully described in a probabilistic sense by a finite number of N carefully selected samples $\{f_i\}_{i=1}^N$ and their associated weights $\{p_i\}_{i=1}^N$ such that $\sum_{i=1}^N p_i = 1$. Therefore, FQ approximates a random function with a "simple function" instead of using parametric representations, such as Karhunmen-Loève expansion or Spectral Representation. Specifically, multi-dimensional FQ considers a random function $F(\xi,\omega)$ defined in the probability space $(\Omega,\mathscr{F},\mathbb{P})$ as $F:\Xi\times\Omega\to\mathbb{R}$, where Ξ is the (multi-dimensional) space domain in $\mathbb{R}^n$, Ω is the sample space, ξ is a point in Ξ and ω is a point in Ω [3].
A random function as defined above could also be interpreted as a random variable $F(\omega)$ with values in the space of square integrable functions: $F:\Omega\to L^2(\Xi)$. Therefore, every outcome ω of the sample space is mapped by the random function to a certain n-dimensional realization in the $L^2(\Xi)$ space.
On the other hand, the random function F_N, which approximates F, is defined as:

$$F_N(\xi,\omega)=\sum_{i=1}^{N} f_i(\xi)\cdot 1_{\Omega_i}(\omega); \qquad 1_{\Omega_i}(\omega)=\begin{cases} 1, & \text{if } \omega\in\Omega_i \\ 0, & \text{otherwise.}\end{cases} \tag{1}$$

where the representative functions $\{f_i(\xi)\}_{i=1}^N$ are called "quanta" and 1_{Ω_i} is the indicator function associated with the event $\Omega_i\subset\Omega$. Therefore, the probability space Ω is partitioned into a mutually exclusive and collectively exhaustive sets $\{\Omega_i\}_{i=1}^N$. Each event Ω_i has an associated probability $\mathbb{P}(\Omega_i)$ and a representative function $f_i(\xi)$, which is called "quantum". Quantum $f_i(\xi)$ represents all the sample functions associated with the ωs belonging to event Ω_i. In summary, the approximation F_N maps all outcomes $\omega\in\Omega_i$ to the same quantum $f_i(\xi)$, whereas the random function F maps each outcome ω to a different realization.
The principal characteristic of FQ is its optimality criterion, which is the mean square convergence of the approximate representation F_N to the actual random function F. Therefore, in principle the quanta $\{f_i(\xi)\}_{i=1}^N$ selected by FQ to represent the various $\{\Omega_i\}_{i=1}^N$ are always going to be optimal (in the mean-square sense) for a specific number N, called "quantizer size."
The same process of partitioning and approximation can also be seen in the space of square integrable functions $L^2(\Xi)$. From this perspective, the $L^2(\Xi)$ space is tasseled into $\{V_i\}_{i=1}^N$, where

each tassel V_i collects all the realizations corresponding to $F(\omega)$ with $\omega \in \Omega_i$. Therefore, FQ induces a tessellation $\{V_i\}_{i=1}^N$ in the $L^2(\Xi)$ space and consequently a corresponding partition $\{\Omega_i\}_{i=1}^N$ of Ω. Based on this partition, the probability $\mathbb{P}(\Omega_i)$ associated with each event is computed. Given the relationship between the two spaces, the probability $\mathbb{P}_F(V_i)$ is equal to the associate $\mathbb{P}(\Omega_i)$.
In order to construct the tessellation $\{V_i\}_{i=1}^N$ and compute the corresponding weights, Miranda and Bocchini [13] developed FQ-IDCVT. This technique is based on the idea of Voronoi Tessellation (VT), which is a process of partitioning a finite-dimensional Euclidean space $\mathbb{R}^n$ into regions $\{T_i\}_{i=1}^N$ called "Voronoi tassels." Each tassel has a generating point and is defined as:

$$T_i = \{\mathbf{y} \in \mathbb{R}^n \quad | \quad \|\mathbf{y} - \hat{\mathbf{y}}_i\| < \|\mathbf{y} - \hat{\mathbf{y}}_j\| \quad \text{for} j = 1, 2, ..., N; j \neq i\} \tag{2}$$

where $\|\cdot\|$ is the Euclidean norm. According to Equation (2), all the points $\mathbf{y} \in \mathbb{R}^n$ that belong to tassel T_i are closer to the generating point $\hat{\mathbf{y}}_i$ than to any other generating point $\hat{\mathbf{y}}_{j\neq i}$. A special case of VT is the Centroidal Voronoi Tessellation (CVT), where each generating point $\hat{\mathbf{y}}_i$, is also the centroid of the tassel T_i and is expressed as $\bar{\mathbf{y}}_i$. Miranda and Bocchini [13] extended the concept of CVT to the infinite-dimensional space of square integrable functions $L^2(\Xi)$, in which case a tassel is defined as follows:

$$T_i = \{F(\omega) \in L^2(\Xi) \quad | \quad \|F(\omega) - \hat{f}_i\|_{L^2(\Xi)} < \|F(\omega) - \hat{f}_j\|_{L^2(\Xi)} \quad \text{for} j = 1, 2, ..., N; j \neq i\} \tag{3}$$

where $\hat{f}_i$ is the generating point of T_i and $\|\cdot\|_{L^2(\Xi)}$ is the $L^2(\Xi)$ norm. Equation (3) denotes that all realizations $F(\omega)$ closer to $\hat{f}_i$ than any other $\hat{f}_{j\neq i}$ are clustered in T_i. In this specific case the generating points $\{\hat{f}_i\}_{i=1}^N$ are also the centroidal points $\{\bar{f}_i\}_{i=1}^N$ of $\{T_i\}_{i=1}^N$. The tassels that are generated to construct a CVT by Equation (3) will be used as the tassels in the FQ sense. Therefore, $T_i \equiv V_i$ and $\hat{f}_i \equiv \bar{f}_i \equiv f_i$.
To impose convergence of the approximate representation F_N to the random function F, the following functional called "distortion" needs to be minimized:

$$\Delta(\{V_i, f_i\}_{i=1}^N) = \sum_{i=1}^N \int_{V_i} \|f(\omega) - f_i\|_{L^2(\Xi)} d\mathbb{P}_F \tag{4}$$

Luschgy and Pagés [11] proved that a global minimizer exists, and Miranda and Bocchini [13] proved that this minimizer has to necessarily correspond to a CVT of $L^2(\Xi)$, that is equivalent to an IDCVT, but may not be unique.
For practical applications, usually a large and representative set of samples of the random functions is considered, and through the construction of the IDCVT, they end up being grouped (or clustered) into N tassels according to the minimum $\|\cdot\|_{L^2(\Xi)}$ norm. This clustering process can be performed in several ways, some of which do not yield a IDCVT. In particular, when a finite number N_{sim} of samples is used to represent the L^2 space, the integral in Equation (4) is replaced by a sum:

$$\Delta(\{V_i, f_i\}_{i=1}^N) = \sum_{i=1}^N \sum_{j=1}^{N_{sim}} \|F(\omega_j) - f_i\|_{L^2(\Xi)} \cdot 1_{\Omega_i}(\omega) \tag{5}$$

The function F and so each of its realizations $F(\omega_j)$ are discretized with a resolution equal to R [3].
Miranda and Bocchini [13] adapted *Lloyd's Algorithm* used in a finite-dimensional Euclidean space and extended it to cluster random functions for FQ-IDCVT. The algorithm proceeds as follows:

1. Randomly selects N out of N_{sim} realizations as the initial quanta $\{f_i\}_{i=1}^{N}$
2. Performs a Voronoi tessellation of the $L^2(\Xi)$ space
3. Updates each quantum with the average of the samples in the cluster
4. Returns to Step 2 until the functional distortion in Equation (5) converges to a stable value

To guarantee that convergence has been met, i.e. Equation (5) has been minimized, the algorithm should stop when in one iteration all realizations remain in the same cluster.

3 Clustering Algorithms

In this section, we will provide first a conceptual overview of the benchmark algorithm (*Lloyd's algorithm with exhaustive search*) and then describe the four alternatives that we have selected and considered for the quantization process. These algorithms were developed in the field of machine learning and data mining in computer science and adapted herein to cluster random functions.

For example, the first alternative preserves the general scheme of *Lloyd's algorithm*, but takes advantage of the principle of triangle inequality to construct the VT [4, 7]. The other alternatives replace completely Lloyd's approach, and they are hierarchical classification tree, k-nearest-neighbor, and k-D Tree nearest-neighbor. There are other clustering techniques that have not been covered in this survey, adopting different machine learning methods, such as Support Vector Clustering, AdaBoost, and Self Organizing Maps [15, 8].

3.1 Lloyd's Algorithm with Exhaustive Search (LX)

The simplest and most accurate way to form a VT of the L^2 space in step 2 of *Lloyd's algorithm* is to perform an exhaustive search. This means that at each iteration of the algorithm, the distances between each realization $F(\omega_j)$ and all quanta $\{f_i\}_{i=1}^{N}$ are computed, and each realization is assigned to the cluster of the closest quantum. Although this basic clustering algorithm is conceptually straightforward, its computational cost is an issue when multi-dimensional fields are considered and the resolution R is large. The complexity of *Lloyd's algorithm with exhaustive search* is $\mathcal{O}(N_{sim} \cdot N \cdot R \cdot T)$, where T is the number of iterations needed to meet convergence. The solution of this operation is not unique because it depends on the initial selection of the random quanta in step 1 [15]. However, the numerical examples presented in [13] show that the choice of initial quanta leads to different solutions but with extremely similar distortion (less than 0.5%). This means that the clustering algorithm may converge to local minima, which are very close to the global minimum.

Arthur and Vassilvitskii proposed a variant called "*k-means*++" which selects an optimal initial set of quanta and allows LX to reduce the number of iterations required to reach convergence [1].

3.2 Lloyd's Algorithm Combined with Elkan's Algorithm (LE)

Elkan's algorithm replaces the exhaustive search in *Lloyd's Algorithm* with a more efficient technique for the construction of the VT (i.e., step 2). It performs much better than LX when either the number of quanta or the resolution of the problem are large [4, 7]. In particular, the algorithm reduces the number of distances calculated by exhaustive search by leveraging the triangular inequality:

$$\|F(\omega_j) - f_m\| \leq \|F(\omega_j) - f_n\| + \|f_n - f_m\| \tag{6}$$

Where $F(\omega_j)$ is a generic realization and f_m, f_n are two generic quanta. From the inequality, it can be proven also that if $\|F(\omega_j)-f_m\| \leq 0.5 \cdot \|f_n - f_m\|$, then it is also true that $\|F(\omega_j)-f_m\| \leq \|F(\omega_j)-f_n\|$ and it is not necessary to compute $\|F(\omega_j)-f_n\|$ [4]. The algorithm proceeds iteratively as *Lloyd's Algorithm*, but before computing the distance between a realization and the updated quanta in the current iteration, it first determines which quanta are candidates to be the closest, using the mentioned inequality and a system of upper- lower-bounds [7]. The complexity of Elkan's algorithm is $\mathcal{O}(N_{sim} \cdot N \cdot R + N^2 \cdot R \cdot T)$, with T being the number of iterations to meet convergence.

3.3 Hierarchical Classification Trees (CT)

A classification tree algorithm searches for similarities among the N_{sim} realizations in $L^2(\Xi)$, and groups them together in one of two possible ways. The first approach is *agglomerative* (bottom-up), which means that each of the N_{sim} realizations represents a cluster itself, and then pairs are iteratively merged to form larger clusters based on the L^2 norm until N clusters are reached. Although several criteria for merging the clusters are available, in this study, the L^2 norm is computed among the centroids of the clusters. The second approach is *divisive* (top-down), which means that the clustering process initially assumes that all the N_{sim} realizations are in one cluster. Then the algorithm separates these elements binarily into subgroups using a threshold, until N subgroups are reached [8].
Most of the CT algorithms are agglomerative [8], and thus only this type of CT is considered in this study. The complexity of the agglomerative CT is $\mathcal{O}((N_{sim}-N)^2 \cdot R)$, which means it requires at most $(N_{sim}-N)^2$ operations times the resolution R to complete the classification of N_{sim} realizations into N clusters. This algorithm becomes slow for large datasets [12].
Even though this algorithm uses the L^2 norm, and represents the clusters using the centroids (i.e., the quanta), it does not attempt to create a CVT of the space, nor do the following two alternatives.

3.4 Exhaustive k-Nearest Neighbor Search (KN)

The basic idea behind the exhaustive k-Nearest Neighbor search algorithm is to evaluate the proximity of each realization with its neighbor quanta. The neighborhood is defined as a hypercube centered at the realization to be classified, and the size of the hypercube expands by increments δ until k quanta are enclosed in the neighborhood. Defining if a quantum belongs to a hypercube can be done simply by looking at inequalities, and it is much faster than computing the distance from the realization. In the exhaustive version of the algorithm, parameter k coincides with N. Once the neighbor quanta have been identified, there are several criteria to assign a realization to one of them. However, in this study, each realization is assigned to the closest quanta in the neighborhood based on the L^2 norm. Also, the initial estimate of the N quanta is computed using an exhaustive search for a random subset of N_{sim} of size equal to $4 \cdot N$. The complexity of this initialization is $\mathcal{O}(N)$, while the complexity of the search algorithm is $\mathcal{O}(N_{sim} \cdot N \cdot R)$. A variant of this technique, combines KN with classification trees in subregions of the domain. One example is the k-D Tree method [2] that follows.

3.5 $\bar{k}$-D Tree (KD)

The $\bar{k}$-D Tree algorithm is a generalization of the binary tree in the $\bar{k}$ dimensional space $\mathbb{R}^{\bar{k}}$ [2]. In our case, parameter $\bar{k}$ is equal to the resolution R of the realizations. The entire space is split into binary cells using hyperplanes perpendicular to selected directions at certain thresholds. The hyperplanes are in general perpendicular to the $\bar{k}$-dimensional axes. For instance, in a 4-D space, the four hyperplanes would be perpendicular to the directions $(1,0,0,0), (0,1,0,0), (0,0,1,0)$, and $(0,0,0,1)$, although other directions can be chosen. The threshold is typically chosen to be the median of the points' coordinates along the axes in each cell, but other approaches are also valid. The splitting operation is repeated recursively until each cell includes one quantum. Then, KN is performed within each cell independently. Usually, each neighbor stops growing when it reaches the first quantum (i.e., parameter k of KN is equal to 1, and it should not be confused with parameter $\bar{k}$ of KD, which is equal to R). The complexity for training the $\bar{k}$-D Tree is $\mathscr{O}(N \cdot logN)$ because in general each visited node of the tree reduces the number of points by half [7]. The searching process has complexity $\mathscr{O}(k \cdot N_{sim} \cdot R)$.

4 Numerical Example

To compare the performance of the algorithms described in the previous section, a 1D non-stationary random process was used as numerical example. All the numerical simulations were performed with the commercial software *MATLAB*® [14] installed on a workstation with 64GB of RAM and two *Intel*® CPUs Xeon E5-2620 with six cores and frequency equal to 2.0GHz. In this example, $3,000$ realizations were generated for this process characterized by the following spectral density function $S_{FF}(\omega)$ and modulating function $A(t)$, depending on the radial frequency ω and time t respectively.

$$S_{FF}(\omega) = \frac{\sigma^2 \cdot b^3 \cdot \omega^2}{4} \cdot \exp\left(-b \cdot |\omega|\right)$$

$$A(t) = \begin{cases} \frac{3 \cdot t}{P} & \text{if} \quad t \leqslant \frac{P}{3} \\ 1 & \text{if} \quad \frac{1}{3} < t \leqslant \frac{3 \cdot P}{5} \\ \exp[-0.02 \cdot (t - \frac{3 \cdot P}{5})] & \text{if} \quad t > \frac{3 \cdot P}{5} \end{cases} \tag{7}$$

The coefficients σ and b are the standard deviation and the correlation parameter of the random process, set equal to 2.0 and 1.0 respectively, while P is a parameter of the modulating function, which depends on the number of points in the frequency domain N_ω and the cutoff frequency ω_u. In this example, P is equal to $\frac{2\pi N_\omega}{\omega_u}$, with $N_\omega = 32$ and $\omega_u = 3.14 \quad rad/s$. A set

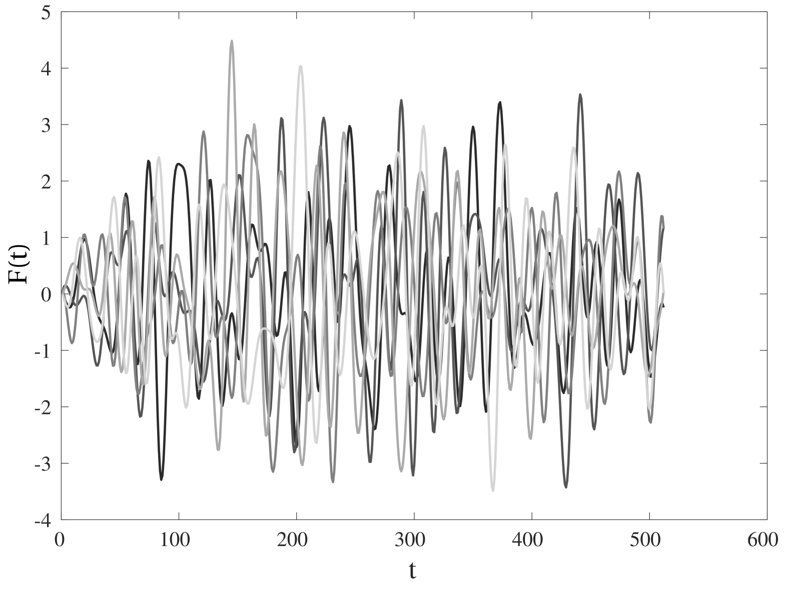

Figure 1: The figure shows five random samples out of $3,000$ *realizations of a uni-dimensional, non-stationary random process with zero mean, standard deviation* $\sigma = 2.0$ *and correlation parameter* $b =$ 1.0.

Table 1: Discretization of the random process.

Type of Process	Parameter R - Random process's discretization size										
1D Non-Stationary	128	256	512	1,024	2,048	4,096	8,192	16,384	32,768	65,536	131,072

of 5 random realizations for this process is shown in Figure 1. FQ-IDCVT was performed on the N_{sim} samples and each sample was discretized with R points in the time domain. This means that as parameter R increases, the numerical approximation of the random function F improves. The drawback is that in such a case the computational cost increases. If a d-dimensional random field with resolution R is analyzed, the computational cost increases proportionally to R^d. In this example, eleven independent cases for the resolution R were considered, as listed in Table 1. In each case, FQ-IDCVT was performed with each of the five algorithms investigated. Four out of five algorithms (i.e., LX, CT, KN, and KD) are standard tools of the *Statistics and Machine Learning Toolbox* of *MATLAB*® [14]. The code for Elkan's classification algorithm was retrieved from [5]. The performance of each algorithm is measured in terms of computational time and relative distortion (*RD*) with respect to LX, which is considered the reference. *RD* was estimated using:

$$RD = \frac{\Delta(\{V_i, f_i\}_{i=1}^{N})_{\mathrm{LX}} - \Delta(\{V_i, f_i\}_{i=1}^{N})_z}{\Delta(\{V_i, f_i\}_{i=1}^{N})_{\mathrm{LX}}} \tag{8}$$

where $z =$ (LE, CT, KN, KD). Figure 2 displays a comparison of the computational time and

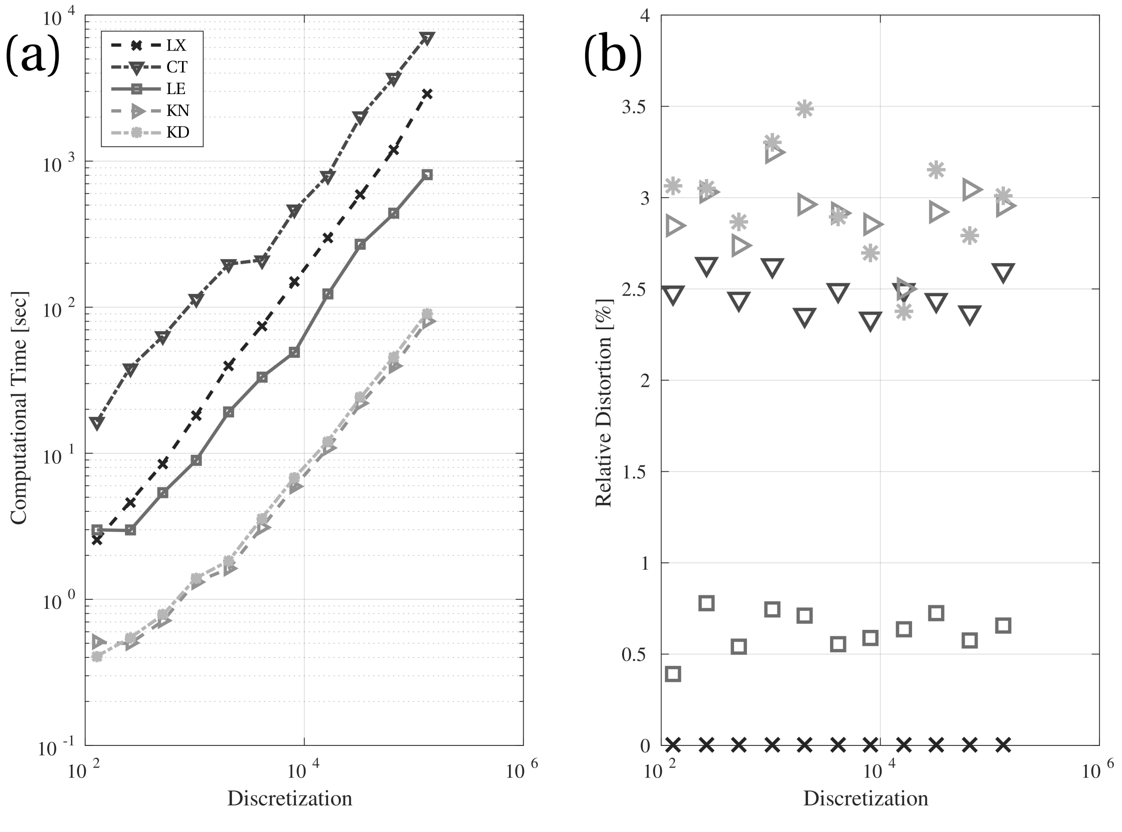

Figure 2: (a) computational time required by each algorithm to perform FQ-IDCVT with 3,000 *samples for different discretization sizes of the random process; (b) relative distortion between Lloyd's algorithm with exhaustive search (*LX*), Lloyd's algorithm with Elkan's method (*LE*), Classification Tree (*CT*), k-Nearest Neighbor (*KN*), and $\bar{k}$-D Tree algorithm (*KD*) for different resolutions.*

the relative distortion among the algorithms and the benchmark solution. Figure 2a shows that the computational cost increases as the discretization size increases, and that the performance among the algorithms varies by almost two orders of magnitude. As expected, for large samples, the CT is the algorithm with the highest computational time, while the fastest algorithms for the quantization are KN, and KD. When KN, and KD are compared to LX, the computational time is about 1.5 orders of magnitude smaller for values of *R* greater than 1,024. For relatively small values of *R*, LE is similar to LX, because the cost of computing distances is comparable to the time that the algorithms requires to check the triangle inequalities with the other quanta.

In terms of constructing the CVT tessellation, because only LX and LE are iterative procedures, they can construct a CVT, while CT, KN, and KD are not iterative and do not construct a CVT. In this example, for CT, KN, and KD the percentage of realizations to be rearranged in order to achieve a CVT is in the range of 15-25%, as shown in Table 2. Finally, the LE algorithm is about 66% faster than LX and about one order of magnitude slower than KN and KD, but it is the most accurate alternative to LX. For instance, Figure 2b shows that despite the size of the parameter *R*, the LE algorithm is always the one that finds quanta with the minimum distortion, and thus the closest to *Lloyd's Algorithm with exhaustive search*. All the other algorithms also provide values of the relative distortion that are not sensitive to the discretization parameter *R*. CT gives values of the relative distortion in the range $2.0 - 2.5\%$ while the relative distortion obtained with KN and KD oscillates around 3.0%.

Figure 3 shows a graphical representation of the computational time versus the relative distortion obtained with all clustering algorithms for a fixed value of the resolution *R* equal to 128; 1,024; 16,384 and 131,072. In each subplot, a vertical line indicates the benchmark solution with respect to time. Algorithms falling to the right of the line are not computationally efficient.

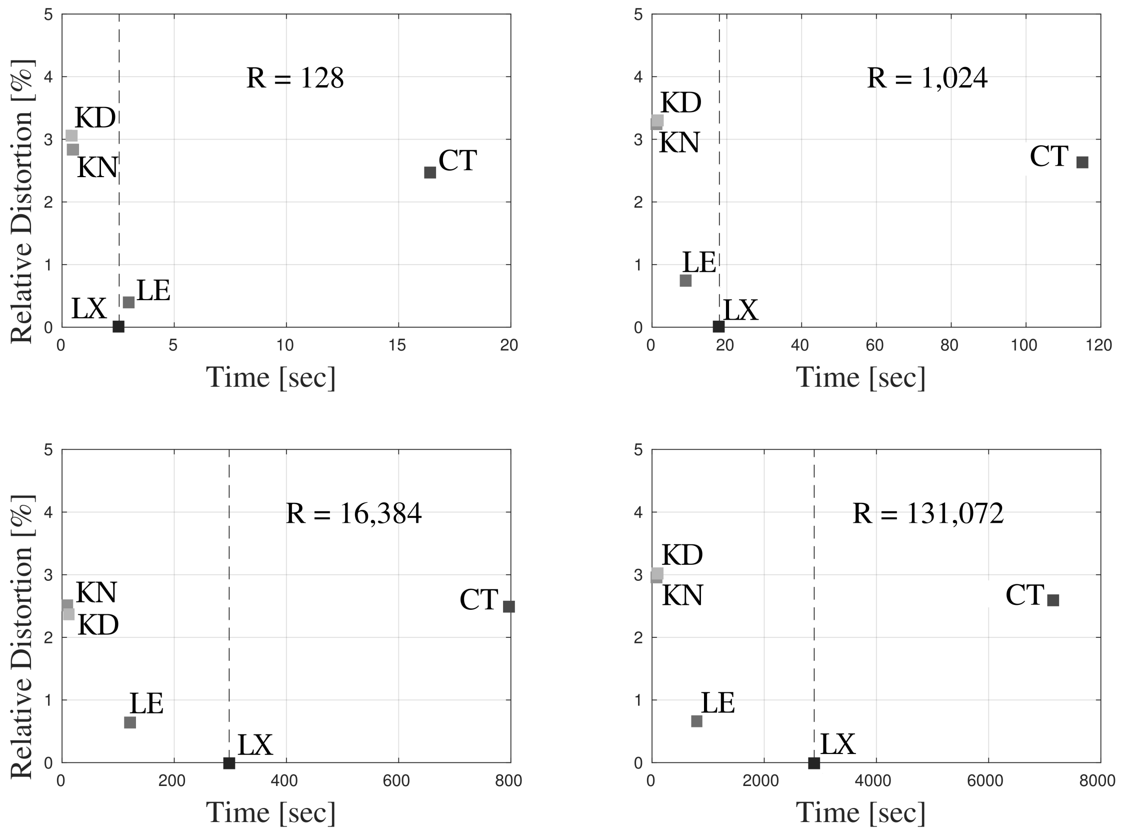

Figure 3: Time vs Relative Distortion for different values of the random process's discretization parameter R = 128, 1,024, 16,384, and 131,072.

Table 2: Summary of results

Algorithm	CVT	CVT Achieved [%]	Comp. Time [sec.]	Rel. Distorion [%]
LX	✓	100.0	[2.5 - 2.9e+3]	N/A
LE	✓	100.0	[3.0 - 0.8e+3]	[0.4 - 0.8]
CT	✗	78.1	[16.4 - 7.1e+3]	[2.3 - 2.6]
KN	✗	86.8	[0.5 - 0.08e+3]	[2.5 - 3.2]
KD	✗	76.6	[0.4 - 0.09e+3]	[2.4 - 3.5]

The best compromise is represented by the point closest to the origin, which turns out to be EL for high resolutions and LX for low resolutions.

5 Conclusion

In this paper the computational efficiency of five clustering algorithms was investigated when FQ-IDCVT of a random function was performed. Among these algorithms, *Lloyd's Algorithm with exhaustive search* was used as the benchmark solution.
The results of the analyses show that among the four algorithms compared to LX, Elkan's algorithm is the most accurate, with a relative distortion equal to 0.6%, while KN, KD, and CT are in the range of $2.3 - 3.5\%$. The lowest value observed in this range is obtained with the classification tree algorithm. On the other hand, KN and KD are faster than LX by 1.5 orders of magnitude while Elkan's algorithm is only 66% faster than the benchmark approach, and the classification tree algorithm is the slowest. According to these results, the clustering algorithm that provides the best trade-off in terms of accuracy and efficiency for the functional quantization of multi-dimensional random field seems to be Elkan's algorithm.

Acknowledgement

The support from Lehigh University through the CORE Project "Functional Quantization of Ebola Zoonotic Spreading" is gratefully acknowledged.

References

[1] D. Arthur and S. Vassilvitskii. "k-means++: the advantages of careful seeding". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007).

[2] S. Arya. "Algorithms for fast vector quantization". In: *Data Compression Conference, 1993.* (1993), pp. 1–17.

[3] V. Christou, P. Bocchini, and M. J. Miranda. "Optimal representation of multi-dimensional random fields with a moderate number of samples: application to stochastic mechanics". In: *Probabilistic Engineering Mechanics* 44 (2016), pp. 53–65.

[4] C. Elkan. "Using the Triangle Inequality to Accelerate k-Means". In: *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)* (2003), pp. 147–153.

[5] *Fast k-means code for Matlab Source File*. `http://cseweb.ucsd.edu/~elkan/fastkmeans.html`. Accessed: 2016-12-20.

[6] G. Fiorillo. "Reliability and Risk Analysis of Bridge Networks Under the Effect of Highway Traffic Load". PhD thesis. The City College of New York, CUNY., 2015.

[7] G. Hamerly. “Making k -means even faster”. In: *2010 SIAM international conference on data mining (SDM 2010)* (2010), pp. 130–140.

[8] G. James et al. *An Introduction to Statistical Learning, with Applications in R*. Springer, 2013.

[9] A. Karamlou and P. Bocchini. “Sequencing algorithm with multiple-input genetic operators: Application to disaster resilience”. In: *Engineering Structures* 117 (2016), pp. 591–602.

[10] S. Lloyd. “Least squares quantization in PCM”. In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137.

[11] H. Luschgy and G. Pagès. “Functional quantization of Gaussian processes”. In: *Journal of Functional Analysis* 196.2 (2002), pp. 486–531.

[12] O. Maimon and L. Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer, 2010.

[13] M. J. Miranda and P. Bocchini. “A versatile technique for the optimal approximation of random processes by Functional Quantization”. In: *Applied Mathematics and Computation* 271 (2015), pp. 935–958.

[14] The Mathworks Inc. *Matlab version 7.8 - R2009a*. Natick, Massachusetts, 2009.

[15] X. Wu et al. *Top 10 algorithms in data mining*. Vol. 14. 1. 2008, pp. 1–37.