An International Journal
computers & mathematics with applications

PERGAMON Computers and Mathematics with Applications 44 (2002) 511–523
www.elsevier.com/locate/camwa

Numerical Studies of MacQueen's k-Means Algorithm for Computing the Centroidal Voronoi Tessellations

QIANG DU*
Department of Mathematics, Penn State University
University Park, PA 16802, U.S.A.
and
Department of Mathematics, Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong, P.R. China
qdu@math.psu.edu

TAK-WIN WONG
Department of Mathematics, Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong, P.R. China

(Received April 2001; accepted May 2001)

Abstract—We study a probabilistic algorithm for the computation of the centroidal Voronoi tessellation which is a Voronoi tessellation of a given set such that the associated generating points are centroids (centers of mass) of the corresponding Voronoi regions. We discuss various issues related to the implementation of the algorithm and provide numerical results. Some measures to improve the performance are also presented. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords—Centroidal Voronoi tessellations, Probabilistic methods, k-means algorithms.

1. INTRODUCTION

The concept of the centroidal Voronoi tessellations (CVTs) has been studied recently in [1]. CVTs have been used in many areas of scientific and engineering applications [1], including data analysis and communication [2], resource allocation [3], grid generation and optimization, and numerical solution of partial differential equations [4].

A number of algorithms for the computation of CVTs have been discussed in [1], including a list of references. Among the known algorithms, the *probabilistic* k-means method by MacQueen is an extremely attractive method for computing the CVTs due to the fact that it relies very little on the underlying geometric structures. In applications where *meshless* computations are desirable, MacQueen's k-means method may become the method of choice. The purpose of this paper is to examine numerically the performance of the k-means method and to identify some possible improvements and extensions of the algorithm.

We would like to thank Dr. G. Basak of HKUST and Prof. M. Gunzburger for helpful discussions.
*This research is supported in part by the State Major Basic Research Grant G199903280 and a grant from the HKRGC. The research was also supported in part by a grant from NSF-CCR.

We first briefly describe the related terminology, see [1] for more detailed discussion. Given a set of input points $\{\mathbf{z}_i\}_{i=1}^k$ belonging to a domain Ω. The *Voronoi* region $\hat{V}_i$ corresponding to the point $\mathbf{z}_i$ consists of all points in the domain that are closer to $\mathbf{z}_i$ than any other point in the set. The set $\{\hat{V}_i\}_{i=1}^k$ forms a partition of ω and is known as a *Voronoi tessellation* or *Voronoi diagram* of Ω. The points $\{\mathbf{z}_i\}_{i=1}^k$ are called *generating points* or *generators*.

A centroidal Voronoi tessellation offers many superior properties than the ordinary Voronoi tessellations and it is associated with a properly defined density function ρ. Given the function ρ defined on a domain Ω, for any region $V \subset \Omega$, the *mass centroid* $\mathbf{z}^*$ of V is

$$\mathbf{z}^* = \frac{\int_V \mathbf{y}\rho(\mathbf{y})\,d\mathbf{y}}{\int_V \rho(\mathbf{y})\,d\mathbf{y}}. \tag{1}$$

Definition 1.1. *Given the set of points $\{\mathbf{z}_i\}_{i=1}^k$ in Ω and a density function ρ, a Voronoi tessellation is called a centroidal Voronoi tessellation (CVT) if*

$$\mathbf{z}_i = \mathbf{z}_i^*, \qquad i = 1, \dots, k, \tag{2}$$

i.e., the points $\mathbf{z}_i$ that serve as generators for the Voronoi regions $\hat{V}_i$ are themselves the mass centroids of those regions.

The mass centers are often also called cluster centers which are simply the *means* of the Voronoi regions with respect to the specified density function. The MacQueen's k-means method refers to the following procedure.

Algorithm 1.1.

```
Input:
  $\Omega$, a closed set, $k$, number of generators, $\rho$, a probability distribution on $\Omega$.
Output:
  CVT with $n$ generators in $\Omega$.
Method:
  1. Generate $n$ random points $\{z_i\}_{i=1}^k$ according to the probability distribution $\rho$.
  2. Initialize indices $j_i = 1$ for all $i = 1, 2 \dots, k$.
  3. Sample randomly a point $x \in \Omega$ according to the probability distribution $\rho$.
  4. Find a $z_i$ that closest to $x$ and update the generator $z_i$ by
         $z_i = \frac{j_i * z_i + x}{j_i + 1}$.
  5. Update the $j_i$ associated with the above $z_i$ by setting $j_i = j_i + 1$.
  6. Repeat procedures 3–5 until a stopping criterion is met.
```

Here, the indices $\{j_i\}$ are used to keep track the number of updates on each generators.

In this paper, we present some numerical studies of MacQueen's k-means algorithm in both one and two-dimensional spaces. The paper is organized as follows, in Section 2, the properties of k-means algorithm and its numerical implementation are discussed. In Section 3, numerical results in both one and two space dimensions are presented. Some analysis of the finding is provided in Section 4.

2. PROPERTIES OF MACQUEEN'S METHOD AND ITS IMPLEMENTATION

An early analysis of the above algorithm was provided in [5], where the almost sure convergence of the energy has been proved. We refer to [1,5–15] for more details.

2.1. Pros and Cons

The obvious advantage of MacQueen's k-means algorithm lies in the fact that it relies very little on the geometric information, except the sorting procedure needed to identify the closest generators. The output of the k-means algorithm does not provide the CVT itself but approximate positions of its generators. Thus, the k-means algorithm is particularly attractive in situations where only the CVT generators are needed.

On the other hand, once the generators are found, there are standard methods to find the Voronoi tessellations. Thus, the k-means method may be complemented by a program to generate the Voronoi tessellations to get a complete construction of the CVT. Empirically, the k-means method saves time by avoiding the construction of the Voronoi regions in the intermediate steps.

For any Voronoi tessellations, one may define the so-called distortion value (or energy, cost, etc.) by the following.

DEFINITION 2.1. *Given the set of points $\{\mathbf{z}_i\}_{i=1}^k$ in Ω, a tessellation $\{\hat{V}_i\}_{i=1}^k$ of Ω and a density function ρ, the functional*

$$\mathcal{E}\left(\left(\mathbf{z}_i,\hat{V}_i\right), i=1,\dots,k\right) = \sum_{i=1}^k \int_{\hat{V}_i} \rho(\mathbf{y})|\mathbf{y}-\mathbf{z}_i|^2\,d\mathbf{y}, \tag{3}$$

is called the distortion value of $(\{\mathbf{z}_i\}_{i=1}^k, \{\hat{V}_i\}_{i=1}^k)$.

The minimum of the distortion value is achieved only when $\{\hat{V}_i\}_{i=1}^k$ forms a CVT with generators $\{\mathbf{z}_i\}_{i=1}^k$. Unlike in a descent type method, the sequence of generators produced by the *probabilistic* k-means method, together with the corresponding Voronoi tessellations, does not necessarily have decreasing distortion values. The fluctuation of the distortion values is certainly a disadvantage of k-means method. This issue will be examined later.

2.2. Quantities Measured in the Implementation

MacQueen's k-means method is very simple to implement, the key ingredients involved are: a random number generator for a given density distribution, a sorting algorithm, an averaging and updating step.

We implemented the method for one-dimensional intervals and two dimension squares using Matlab. To evaluate the algorithm, in the experiments, we measure several quantities that include the number of sampling points, the computational operation count, the distortion value, the error, the relative movement of generators, and the positions of the resulting generators. The operation count, abbreviated as *count* in this paper, denotes the number of multiplications involved when the program is executed. It is proportional to the computational time. We have ignored the effort spent on the sorting procedure at each step as the total time on sorting for each new sampling point is almost fixed with the given number of sampling points.

In order to measure the progress of the k-means method, the change value is used to record the relative movement of the generators and for two sets of generators $\{z_i\}_{i=1}^n$ and $\{z_i'\}_{i=1}^n$, it is given by the Euclidean norm of $\{z_i - z_i'\}_{i=1}^n$ in R^{dn} where d is the space dimension

$$\hat{e}\left(\{z_i\}_{i=1}^n, \{z_i'\}_{i=1}^n\right) = \left(\sum_{i=1}^n |z_i - z_i'|^2\right)^{1/2}. \tag{4}$$

We will replace the change value between consecutive sets of the generators to the change value $\hat{e}(\{z_i\}_{i=1}^n, \{z_i*\}_{i=1}^n)$ between our numerically computed generators $\{z_i\}_{i=1}^n$ and the exact solution $\{z_i*\}_{i=1}^n$ whenever the latter is known.

The measured quantities are used to study the efficiency of the k-means method and the measurements are normally taken as the averaged values of the corresponding quantities in five different realizations (seeded with different random numbers). The results are presented using the averaged values, unless otherwise stated.

2.3. Generating a Random Point According to a Density Function

For certain well-known distributions, one may use the system built-in random number generators. For general nonuniform density distributions, it is possible to apply appropriate transformations [16]. For instance, to generate a random variable x between a and b, with density distribution $\rho(x)$, where $\rho(x) \geq 0$ for any point in the given set $\Omega = [a,b]$, we may define a mapping $F : x \longrightarrow u$ by

$$u = \frac{\int_a^x \rho(x')\, dx'}{\int_a^b \rho(x')\, dx'} = F(x). \tag{5}$$

The mapping is one to one and onto and there exist an inverse of F, namely, F^{-1}. Let a random variable $u \sim U(0,1)$, where $U(0,1)$ represent the uniform distribution with the domain $[0,1]$, one may map u to x by the inverse of equation (5). The random variable x generated this way has ρ as its probability density distribution.

For two-dimensional domains such as

$$\Omega = \{(x,y) \mid f_1(x) \leq y \leq f_2(x), \text{ for } a \leq x \leq b\},$$

where f_1 and f_2 are two smooth functions such that $f_1(x) < f_2(x)$ for all $x \in [a,b]$, a random variable with given density distribution ρ can be generated by first generating its x-coordinate, then its y-coordinate. Again, assuming that $\rho(x,y) > 0$ for any $(x,y) \in \Omega$, one may first generate a random number $u_1 \sim U(0,1)$. Then, one computes x such that

$$u_1 = \frac{\int_a^x \int_{f_1(x')}^{f_2(x')} \rho(x',y')\, dy'\, dx'}{\int_a^b \int_{f_1(x')}^{f_2(x')} \rho(x',y')\, dy'\, dx'} = F(x). \tag{6}$$

That is $x = F^{-1}(u_1)$. Subsequently, with given x, one generates $u_2 \sim U(0,1)$, and the y coordinates is found by

$$u_2 = \frac{\int_{f_1(x)}^{y} \rho(x,y')\, dy'}{\int_{f_1(x)}^{f_2(x)} \rho(x,y')\, dy'} = G(y). \tag{7}$$

That is $y = G^{-1}(u_2)$. One may check that the random number generated this way has the given density distribution ρ.

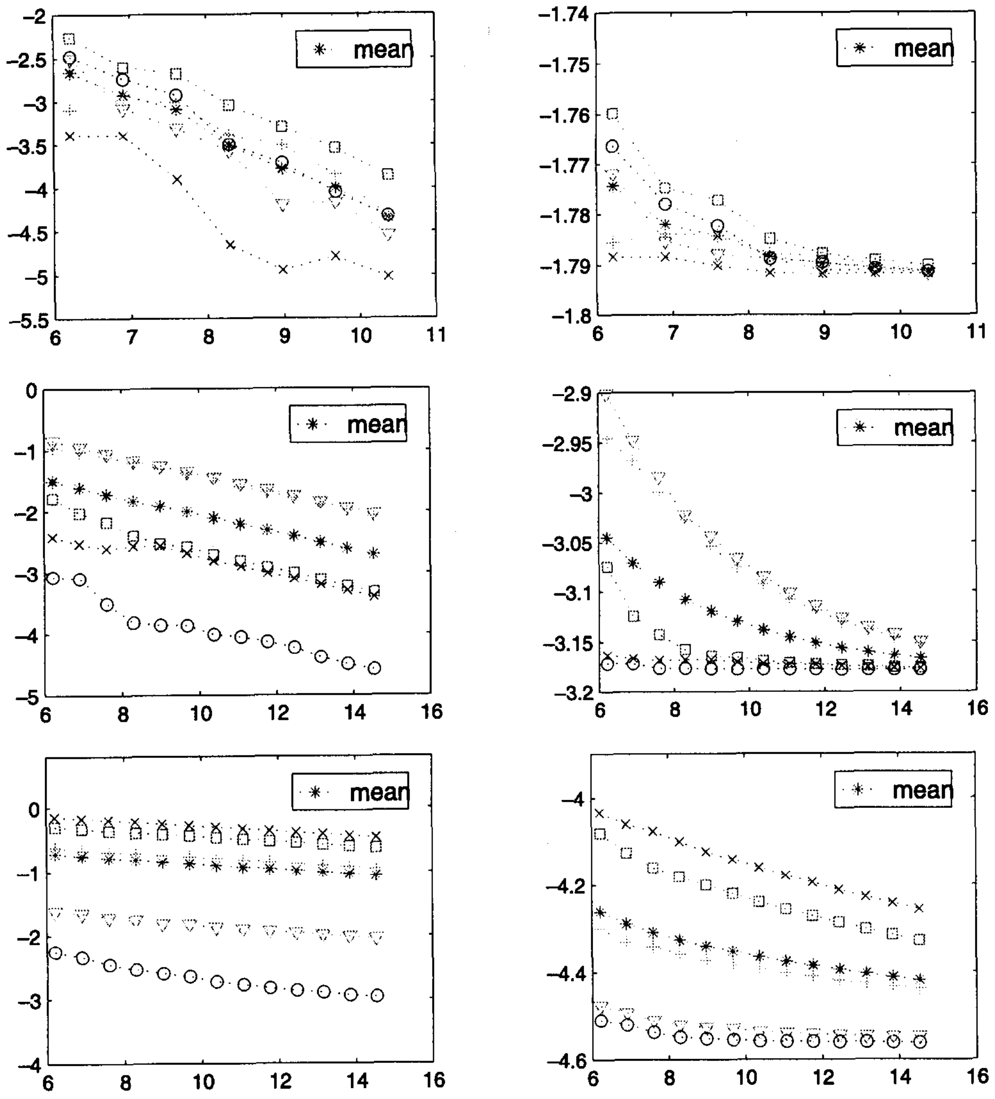
In cases where equations (5)–(7) cannot be solved explicitly, numerical root finding is necessary. When computing the x-coordinate and y-coordinate in equations (5)–(7), one may use adaptive numerical integration rules to compute the single or double integrals. Numerical integrations are also necessary in the computation of the distortion value defined by (3). When it is difficult to solve equations (5)–(7), one may use other methods to generate points with a given distribution such as rejection methods [16].

2.4. Generating the Voronoi Regions

For one-dimensional intervals, the computation of the Voronoi regions for a given set of generators is trivial as the boundary of the Voronoi intervals are the bisectors of neighboring generators. For two-dimensional boxes, we have modified the Matlab built-in functions *voronoi* to suit our needs of computing the Voronoi tessellations. This built-in program is based on the algorithm given in [17].

2.5. Stopping Criterion

When Algorithm 1.1 converges, as the number of sampling points p tends to infinity, the difference between consecutive sets of generators tends to zero, that is, the change value $\hat{e}$ is



(a) log($\hat{e}$) against log(p). (b) log($\mathcal{E}$) against log(p).

Figure 1. Plots for 2, 4, and 8 generators in 1-D with $\rho \equiv 1$.

approaching zero. This remains true when $\hat{e}$ represents the difference between the computed generators and the exact generators of the CVT when the latter is known.

To apply a stopping criterion for terminating the program, we may either set limit for the total number of sampling points p or set tolerance for the change value $\hat{e}$ (either between two consecutive sets of generators $\{z_i'\}_{i=1}^n$ and $\{z_i\}_{i=1}^n$ or between the approximate generators and the exact generators).

Both the limit and the tolerance are taken to be proportional to the size of the domain (the length of the interval in one-dimensional case and the area of the region in two-dimensional case). They are also taken to be proportional to the number of generators k.

In using the change value between consecutive steps as the termination condition, we often replace the change value of a single step by the averaged change values over several, say k, consecutive steps. This is to avoid premature termination of the algorithm should an unlikely event occur so that a new sampling point either coincides with or is too close to an existing generator.

3. NUMERICAL RESULTS

In order to study the numerical convergence of MacQueen's k-means method, the results with different numbers of sampling points generated are presented. We begin with the one-dimensional case, then move to the two-dimensional case. Both uniform and nonuniform densities are used in the test runs.

3.1. One-Dimensional Case

We take the interval $[-1, 1]$ as the domain Ω. The results for the uniform density case are presented first. Then we present some cases with nonuniform densities.

The uniform density

We begin with the case of uniform density in one-dimensional spaces. For simplicity, we take $\Omega = [0, 1]$. The graph of $\log(\hat{e})$ against $\log(p)$ and the graph of $\log(\mathcal{E})$ against $\log(p)$ are plotted in Figure 1a, where p is the number of sampling points generated and $k = 2, 4, 8$ are the numbers of generators corresponding to the pictures in the three rows, respectively.

In Figure 1, the five different realizations are plotted with the symbols $\square$, $+$, $\bigcirc$, $\triangledown$, $\times$, and the mean values of the sets of data (values of $\mathcal{E}$ and $\hat{e}$) are denoted by the symbol $*$.

It can be seen from Figure 1b that the distortion values converge to approximately the same value in all realizations for $k = 2$ and $k = 8$. In fact, the generators all converge to the same CVT. The convergence seems much slower for the case $k = 8$.

According to the numerical data, the slope of the linear least square fitting line of the mean value errors in Figure 1a is approximately -0.41 for $k = 2$, -0.14 for $k = 4$, and -0.042 for $k = 8$. This clearly indicates the reduction in convergence rate as the number of generating points increases.

Nonuniform densities

We take $\rho(x) = x + 1$ first and present the cases of 2, 4, and 8 generators.

Next, we present the results using a normal distribution $\rho(x) = \exp(-x^2/2\sigma^2)/(\sqrt{2\pi}\,\sigma)$ with $\sigma = 0.2$, restricted to the interval $[-1, 1]$.

Just as for the constant uniform density case, the performance becomes poorer for nonuniform densities with a larger number of generators.

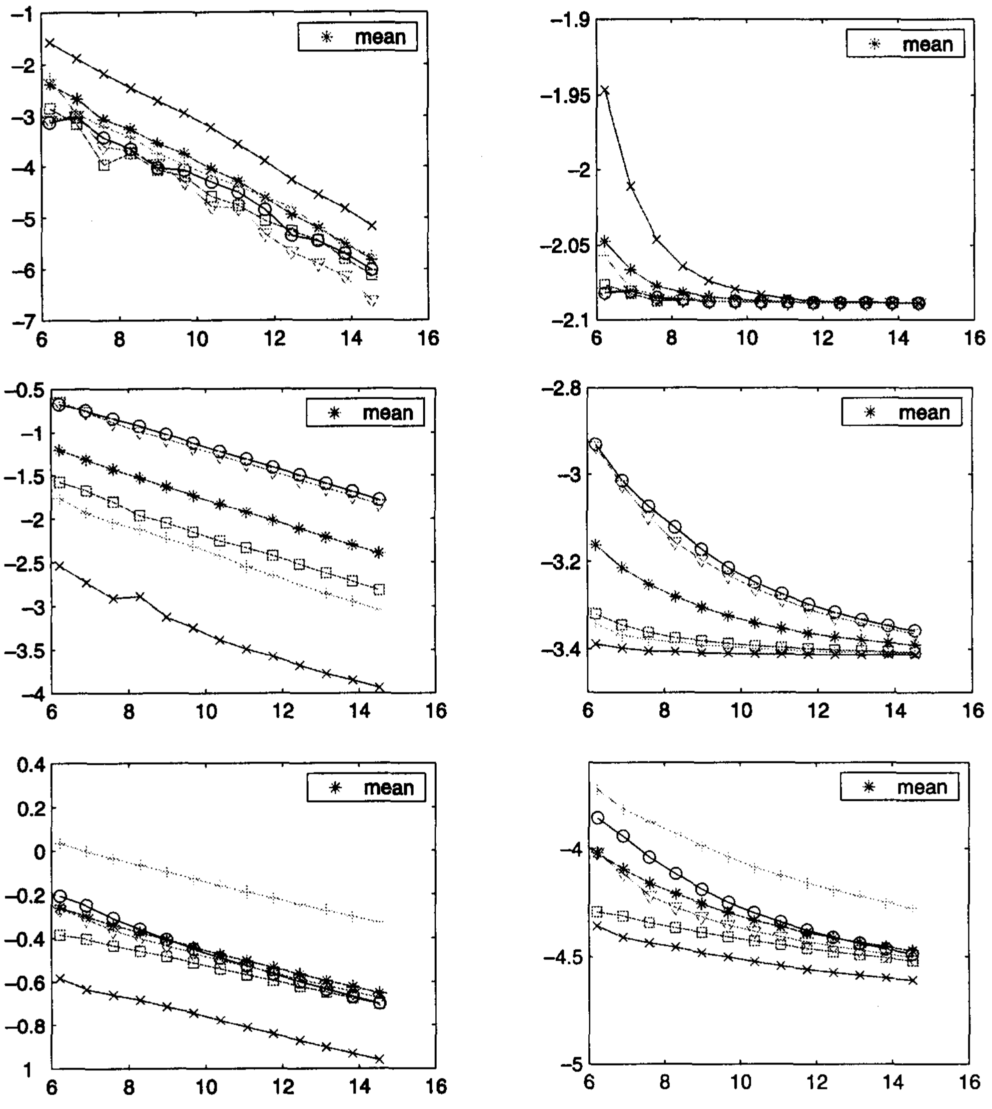
3.2. Two-Dimensional Case

Now, the numerical results in two-dimensional case are studied. The grouping of results is similar to that for the one-dimensional cases where the uniform density cases are presented first and followed by results for the nonuniform densities.

The uniform density

In a square $[-1, 1]^2$, the CVTs corresponding to exact global minimizers of the distortion value have $(0.0, \pm 0.5)$ as generators (for $k = 2$ up to a 90 degree rotation), and have $(\pm 0.25, \pm 0.25)$ as generators (for $k = 4$).

Figure 4 shows $\log(\hat{e})$ as function of $\log(p)$ and $\log(\mathcal{E})$ as a function of $\log(p)$ for $k = 2, 4$ generators with uniform density. The graphs include five sets of data and the set of mean values of the data that again are denoted by the symbol $*$. It can be seen that the distortion value converges to approximately the same value but it may take a longer time for the approximation to converge if the number of generators is increased.



(a) log($\hat{e}$) against log(p).

(b) log($\mathcal{E}$) against log(p).

Figure 2. Plot for 2, 4, 8 generators in 1-D with $\rho(x) = x + 1$.

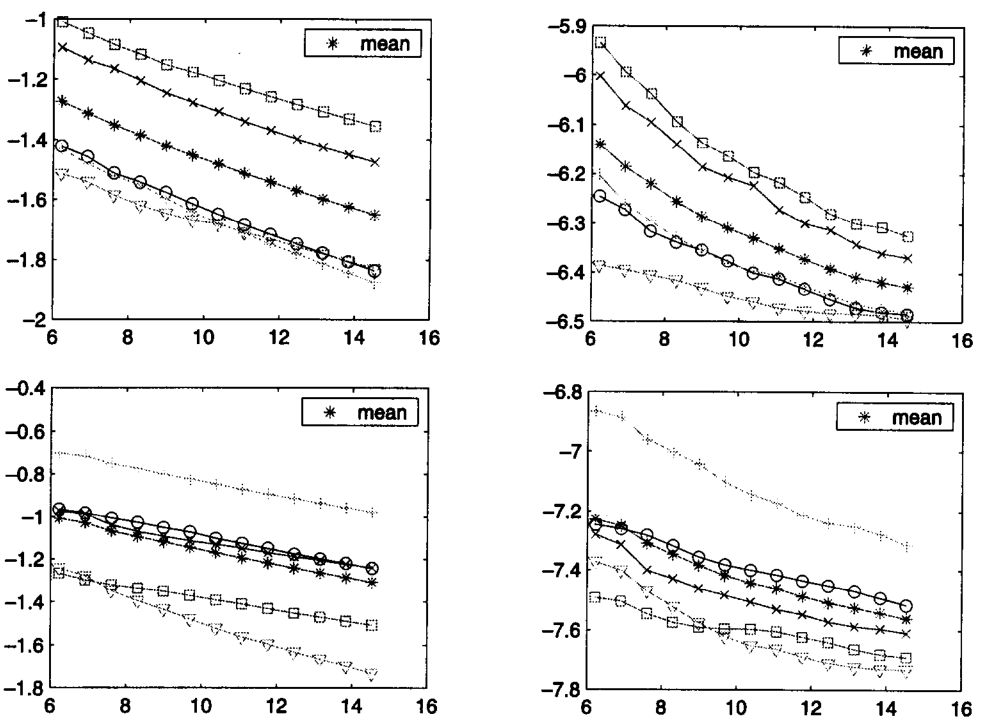
Nonuniform densities

We take a normal distribution $\rho_n(x, y) = \exp(-x^2 - y^2/2\sigma^2)/(2\pi\sigma)$ with $\sigma = 0.2$ in the test run. The results are given in Figure 5. The performance and the error behavior again are similar to the constant density case.

Finally, we present some computed CVTs corresponding to the truncated normal distribution given here.

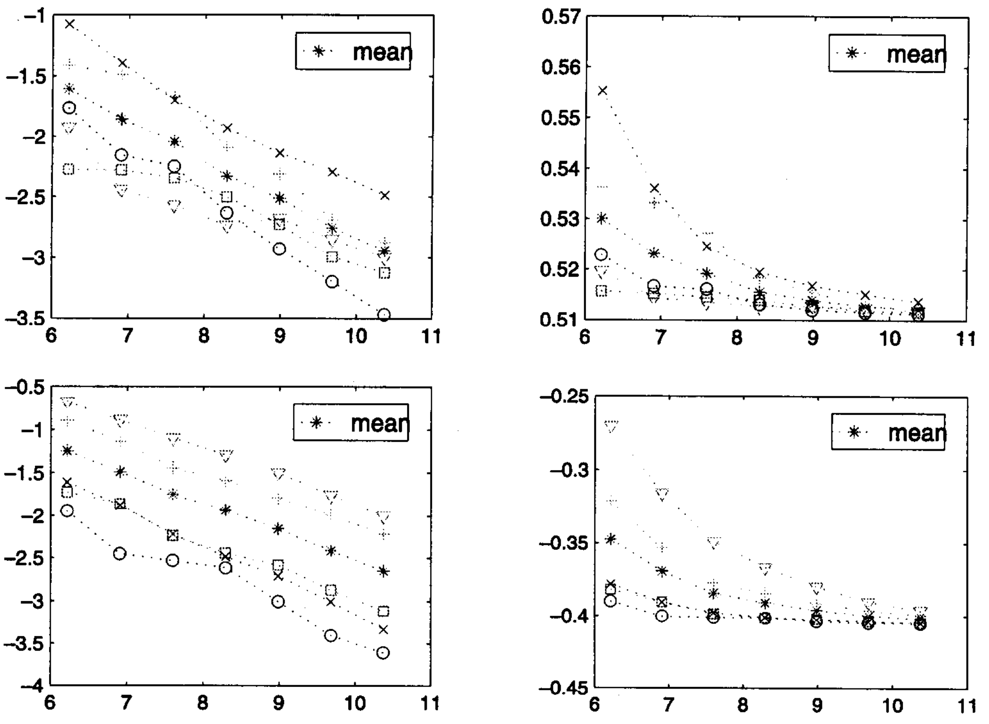
3.3. Discussion

In general, we have found that, in all the figures regarding the convergence behavior, the errors are asymptotically decreasing and the corresponding distortion values have a trend to converge to a same value. The convergence, however, appears to be very slow. Notice that effectively, the x-axis in the figures of errors represents the number of times the sampling points are doubled. By comparison, it takes about 64 fold of the time for generating the data in Figure 1 for $k = 4$



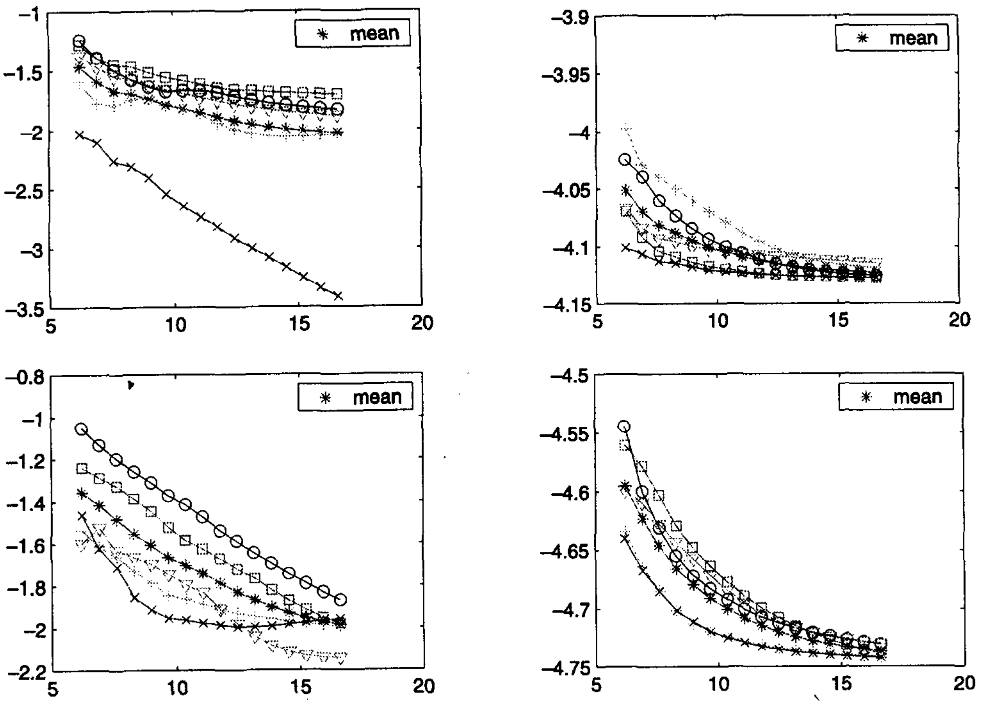
(a) log($\hat{e}$) against log(p) (b) log($\mathcal{E}$) against log(p).

Figure 3. Plot for 8 and 16 generators in 1-D with $\rho(x) = \exp(-x^2/2\sigma^2)/(\sqrt{2\pi}\sigma)$, $\sigma = 0.2$.



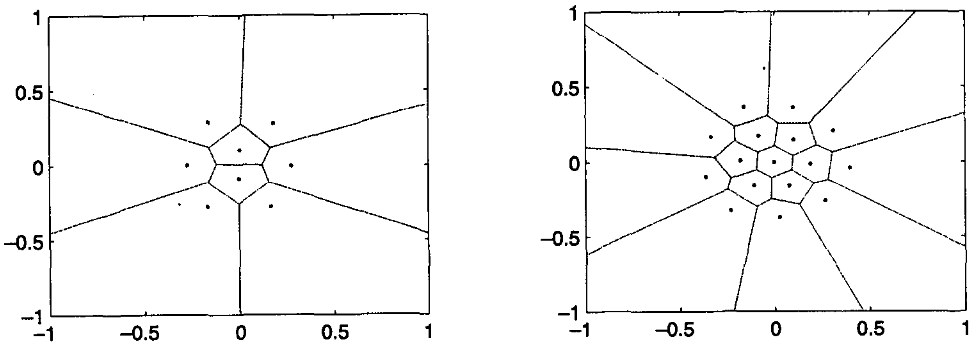
(a) log($\hat{e}$) against log(p). (b) log($\mathcal{E}$) against log(p).

Figure 4. Plot for 2 and 4 generators in 2-D with $\rho \equiv 1$.



(a) $\log(\hat{e})$ against $\log(p)$. (b) $\log(\mathcal{E})$ against $\log(p)$.

Figure 5. Plot for 8 and 16 generators in 2-D with density ρ_n and $\sigma = 0.2$.



(a) (b)

Figure 6. Final 2-D CVTs for 8 (a) and 16 (b) generators with density ρ_n.

than that for $k = 2$. Obviously, much more computation is needed for the $k = 8$ case in the same figure to reach the same precision.

To state in more detail how the distortion value $\mathcal{E}$ and the error of the set of generators $\hat{e}$ changes with respect to the number of sampling points p, we present some data for the two generator case with one-dimensional uniform density in the following table. Here, *count* denotes the operations count.

From Table 1, we see that the operation count is proportional to the number of sampling points, in turn, we can conclude that the computational time is proportional to the number of sampling points generated.

Now consider the case where the number of sampling points generated is fixed and the positions of generators is found numerically for different number of generators. For the uniform density in one dimension, after taking the mean of five sets of different test runs, the graph of log(*error*)

Table 1. Voronoi tessellation of two generators in 1-D with $\rho \equiv 1$.

p	$\mathcal{E}$	$\hat{e}$	*count*
500	0.16959206029287	0.06970582176096	98191
1000	0.16829247045269	0.05404046632319	194591
2000	0.16791010169557	0.04572861680815	390791
4000	0.16722884553248	0.03001874273763	775541
8000	0.16701562397405	0.02280094376283	1549441
16000	0.16687543305963	0.01837471753681	3077491
32000	0.16675634072376	0.01298418433446	6192191

against $\log(k)$ is shown in Figure 7. Using a linear least square fit to the log of the error, we find that error is proportional to $k^{1.0511}$, which roughly justifies our choice of taking the tolerance to be proportional to k in the stopping criteria when the limit for the number of sampling points is not specified in advance. Similar behavior is also observed for other choices of density functions.

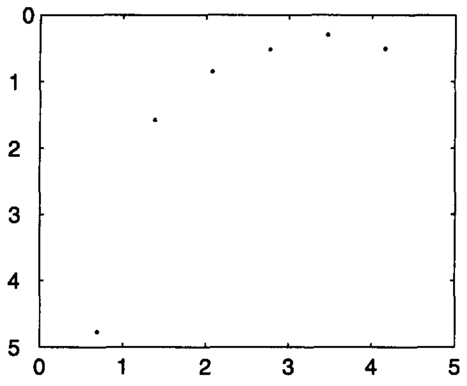


Figure 7. Graph of $\log(\hat{e})$ against $\log(k)$ in 1-D with $\rho \equiv 1$ and 16000 sampling points.

4. FURTHER ANALYSIS AND CONCLUSION

We now present some further analysis and modifications of the algorithms based on our numerical experiments.

4.1. Fluctuation of Distortion Value

First, in the figures for the distortion values, only the values at a few selected iterations are shown. Though they appear to be decreasing, the distortion value of MacQueen's k-means method may actually fluctuate during the iteration. In other words, the final distortion value may not be the minimum among the distortion values corresponding to earlier iterations. In order to obtain a better result, one may record earlier sampling points so that in a post-processing step, the generators with the minimum distortion value among all data sets in the record can be computed. If the distortion value is computed after each update, then only the latest generating set with the smallest distortion value needs to be saved. The details are given in Algorithm 4.1.

For example, consider a CVT of two generators in $[-1, 1]$ with $\rho \equiv 1$, the results from a test run is given in Table 2. 7000 sampling points are generated. The numerical approximations to the exact generators are obtained using Algorithms 1.1 and 4.1, respectively, with 7000 sampling points being generated.

ALGORITHM 4.1.

```
Input:
  $\Omega$, a closed set, $k$, number of generators, $\rho$, a probability distribution on $\Omega$.
Output:
  CVT with $k$ generators in $\Omega$.
Method:
  1. Generate $k$ random point $\{z_i\}_{i=1}^k$ according to the probability distribution $\rho$.
  2. Initialize indices $\{j_i = 1\}_{i=1}^k$.
  3. Set $\{y_i = z_i\}_{i=1}^k$ and compute the distortion value $\hat{e}$ for $\{y_i\}_{i=1}^k\}$.
  4. Sample randomly a point $x \in \Omega$ according to the probability distribution $\rho$.
  5. Find a $z_i$ that closest to $x$ and update the generator $z_i$ by
         $z_i = \frac{(j_i * z_i + x)}{(j_i + 1)}$.
  6. Update the $j_i$ associated with the above $z_i$ by setting $j_i = j_i + 1$.
  7. Compute the new distortion value, if smaller than $\hat{e}$, set $\{y_i = z_i\}_{i=1}^k$.
  8. Repeat procedures 4–7 until a stopping criterion is met. Output $\{y_i\}_{i=1}^k$.
```

Table 2. Positions for the initial, approximate, and exact generators.

Initial	[−0.1669, 0.6618]
Algorithm 1.1	[−0.50260415517905, 0.51649374083476]
Algorithm 4.1	[−0.50156485284546, 0.51317628028480]
Exact	[−0.5, 0.5]

The distortion value is 0.16676989072806 for the pair of generators obtained by Algorithm 4.1 which is smaller than the one obtained by Algorithm 1.1, even though the starting generators and the random sampling points are the same.

Computing the distortion value requires computational overhead, in fact, it relies on the computation of the CVT. Thus, it should not be frequently computed to preserve the computational efficiency.

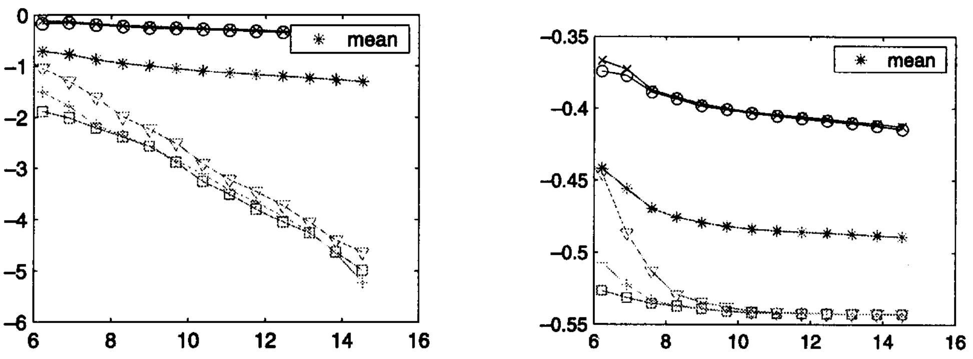
4.2. Random Number Generators

Obviously, generating the sampling points for given density distributions is the major part of the computation in the k-means algorithm. Computationally, one would like to use the most efficient random number generators. There are built-in programs as well as the methods of transformation and rejection. It remains to be explored which is most effective in the computation for specific densities or for a class of density distribution.

4.3. The Effect of the Initial Set of Sampling Points

It is worthy mentioning the significant impact of initial sampling set on the performance of the k-means algorithm. According to Figure 1 for the uniform density in one dimension, the distortion value of the data denoted by $\triangledown$ for $k = 4$ is not as good as the initial distortion value of the data denoted by $\bigcirc$ with the same number of sampling points, and the same is true in for the run denoted by $\times$ for $k = 8$ in comparison with the run denoted by the $\bigcirc$. Similarly, in Figure 8, the test results for the two-dimensional case with $\rho(x, y) = x + 1$ and $k = 4$ is provided, the disparity in performance is also quite evident among the different test runs.

It is easy to see that the good initial generators can lead to a better result without the need of generating a huge number of sampling points. Therefore, this raises the interesting issue of improving the performance by perhaps running multiple copies of the k-means algorithms by



(a) $\log(\hat{e})$ against $\log(p)$. (b) $\log(\mathcal{E})$ against $\log(p)$.

Figure 8. Plot for 4 generators in 2-D with density $\rho(x, y) = x + 1$.

sampling different initial generating sets and sorting out the best answer (or taking an average of all computed answers). Such an idea is ideally suited for parallel computation, a topic that is under further investigation.

Another possible improvement is to introduce an *acceptance rule* for the new sampling points. It is observed that occasional *outliers* may degrade the data sets, though statistically speaking, they should not appear often, thus, one may decide with certain probability whether to accept a new sampling point in the k-means algorithm.

5. CONCLUSION

We have discussed the performance of the k-means method by MacQueen for computing the CVTs through numerical experiments and identified some key factors affecting the performance. Overall, though attractive due to its simplicity, the convergence of the method is very slow. Thus, it is worthwhile to study various accelerations of the k-means method as well as the parallelizations in the future when computing the CVTs with large number of generators.

REFERENCES

1. Q. Du, V. Faber and M. Gunzburger, Centroidal Voronoi tessellations: Applications and algorithms, *SIAM Review* **41**, 637–676, (1999).
2. R. Gray and D. Neuhoff, Quantization, *IEEE Trans. Inform. Theory* **44**, 2325–2383, (1998).
3. M. Iri, K. Murota and T. Ohya, A fast Voronoi-diagram algorithm with applications to geographical optimization problems, *Proceedings of the 11th IFIP Conference on System Modeling and Optimization, Lecture Notes in Control and Information Sciences* **59**, Springer, 273–288, (1984).
4. Q. Du and M. Gunzburger, Grid generation and optimization based on the centroidal Voronoi tessellations, *SIAM J. Sci. Comp.*, (Submitted).
5. J. MacQueen, Some methods for classification and analysis of multivariate observations, In *Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume I*, (Edited by L. Le Cam and J. Neyman), pp. 281–297, University of California, (1967).
6. E. Abaya and F. Wise, Convergence of vector quantizers with applications to optimal quantization, *SIAM J. Appl. Math.* **44**, 183–189, (1984).
7. G. Celeux and G. Govaert, A classification EM algorithm for clustering and two stochastic versions, *Comp. Stat. Data. Anal.* **14**, 315–332, (1992).
8. R. England and D. Benyon, A remark on Algorithm AS136: k-means clustering algorithm, *Appl. Stat.* **30**, 355–356, (1981).
9. J. Hartigan, *Clustering Algorithms*, Wiley-Interscience, New York, (1975).
10. J. Hartigan, Asymptotic distribution for clustering centers, *Ann. Stat.* **6**, 117–131, (1978).
11. J. Hartigan and M. Wong, A k-means clustering algorithm, *Appl. Stat.* **28**, 100–108, (1979).
12. M. Jhun, Bootstrapping k-means clustering, *J. Japanese Soc. Comput. Stat.* **3**, 1–14, (1990).
13. K. Parna, On the stability of k-means clustering in metric spaces, *Tartu Riikl. Ul. Toimetised* **798**, 19–36, (1988).
14. D. Pollard, Strong consistency of k-mean clustering, *Ann. Stat.* **9**, 135–140, (1981).
15. T. Wong, Centroidal Voronoi diagrams and their applications, M. Phil. Thesis, HKUST, Hong Kong, (1999).
16. S. Ross, *A First Course in Probability*, 5th Edition, Prentice-Hall, (1998).

17. S. Fortune, Voronoi diagrams and Delaunay triangulations, In *Computing in Euclidean Geometry*, pp. 193–233, World Sci. Publishing, River Edge, NJ, (1992).
18. A. Okabe, B. Boots and K. Sugihara, *Spatial Tessellations; Concepts and Applications of Voronoi Diagrams*, Wiley, Chichester, (1992).
19. K. Parna, Strong consistency of k-means clustering criterion in separable metric spaces, *Tartu Riikl. Ul. Toimetised* **733**, 86–96, (1986).
20. D. Pollard, A central limit theorem for k-means clustering, *Ann. Stat.* **10**, 919–926, (1982).
21. W. Stute and L. Zhu, Asymptotics of k-means clustering based on projection pursuit, *Sankhya Ser. A* **57**, 462–471, (1995).
22. M. Wong, Asymptotic properties of univariate sample k-means clusters, *J. Classif.* **1**, 255–270, (1984).