

Centroidal Voronoi Tessellations Are Not Good
Jigsaw Puzzles

M. Roger Brauerman; Harvard University
Sarah Joy Zoll; Wellesley College
Christopher L Farmer; Northwest Missouri State University
Dr. Max Gunzburger, Research Advisor; Iowa State University

30 July 1999

1 Introduction

For centuries, bees have known the secrets of the centroidal Voronoi tessellation. This knowledge has allowed them to design the beehive, one of the most efficient structures known to man. The hexagonal architecture provides worker bees with the shortest average distance between cells, thus minimizing travel time and energy expended.

Humans have taken a cue from the crafty bees, and we are now exploring the various applications of centroidal Voronoi tessellations. These applications range from finding the optimal placement of transmitter towers to compressing image data, from creating astronomical theories to solving partial differentiation equations.

As in all fields of study, a basic knowledge of centroidal Voronoi tessellations must be obtained before serious research may be initiated. This knowledge should include the mathematical definition of a centroidal Voronoi tessellation; an understanding of the methods for computing centroidal Voronoi tessellations; and tricks in manipulating the shape, size, and position of centroidal Voronoi regions.

2 Mathematical Significance

Definition

The formal definition of the Voronoi set V_i is as follows:

$$V_i = \{w, z_i, z_j \in \Omega | d(w, z_i) < d(w, z_j), i = 1, 2, \dots, K; i \neq j\}$$

where Ω is an open set of R^n and d is a function of distance.

A centroidal Voronoi region is the special case in which z_i is also the mass centroid of the Voronoi region. The mass centroid c_i of the region V_i is calculated as

$$c_i = \frac{\int_{V_i} x \rho(x) dx}{\int_{V_i} \rho(x) dx}$$

where $\rho(x)$ is the density function of V_i .

Minimizing the Energy Function

The mathematical importance of the centroidal Voronoi tessellation lies in its relationship with the energy function

$$E(z, V) = \sum_{i=1}^n \int_{V_i} \rho(x) |x - z_i|^2 dx$$

where $V \in \Omega$ and $z \in V$. Since $|x - z_i|^2$ is a function of distance, one can assume that Voronoi regions will have minimal energy. This, however, is merely the tip of the iceberg.

Theorem 1 *The energy function is minimized at the mass centroid of a given region.*

Proof.

$$\begin{aligned} E(z, V) &= \sum_{i=1}^n \int_{V_i} \rho(x) |x - z_i|^2 dx, \\ \frac{dE}{dz_i} &= 2 \sum_{i=1}^K \int_{V_i} \rho(x) (x - \hat{z}_i) dx = 0, \\ 2 \sum_{i=1}^K \int_{V_i} x \rho(x) dx - 2 \hat{z}_i \sum_{i=1}^K \int_{V_i} \rho(x) dx &= 0, \\ \hat{z}_i &= \frac{\int_{V_i} x \rho(x) dx}{\int_{V_i} \rho(x) dx} \end{aligned}$$

There is a local minimum of the energy function at $E(\hat{z}_i, V)$ as $|\hat{x}_i - x_i| = 0$. Therefore, the derivative of E with respect to \hat{z} is equal to zero. If we expand the summation and solve for \hat{z}_i , we see that \hat{z}_i is the mass centroid of V_i . ♣

Theorem 2 *For a given set of centers $Z = \{z_i\}$, the energy function $E(Z, V)$ is minimized when V is a Voronoi tessellation.*

Proof. Define the "address" function $z_V : \Omega \rightarrow \{1, 2, \dots, K\}$ as $z_V(x) = z_i$ such that $x \in V_i$. In other words, $z_V(x)$ is the center of the region containing x . Let \hat{V} be a Voronoi tessellation, and let V be another tessellation. The energy function can be written as $\int_{\Omega} f_V(x) dx = \int_{\Omega} \rho(x) |x - z_V(x)|^2 dx$. But by the definition of a Voronoi tessellation, $d(x - z_{\hat{V}}(x)) < d(x - z_i)$, for all z_i . Hence $f_{\hat{V}}(x) \leq f_V(x)$ for all $x \in \Omega$ and so $\int_{\Omega} f_{\hat{V}}(x) dx \leq \int_{\Omega} f_V(x) dx$. Thus, the Voronoi tessellation yields lower energy than any other. ♣

Corollary 1 *Given a tessellation (Z, V) , if \hat{Z} is the ordered set $\{z_i | z_i \text{ is the centroid of } V_i\}$, and (\hat{Z}, \hat{V}) is a Voronoi tessellation for \hat{Z} , then the tessellation $E(Z, V) \geq E(\hat{Z}, \hat{V}) \geq (Z, V)$.*

Corollary 1 is the basis for Lloyd's method, the topic of Section 3.2 for finding a local minimum of E .

3 Computational Algorithms

There are two algorithms commonly used for calculating the centroidal Voronoi regions: K-means and Lloyd's method.

3.1 K-Means

Given a region Ω and a density function $\rho(x)$,

1. Create the probability distribution $P(x)$ associated with $\rho(x)$.
2. Using a Monte Carlo method, chose some initial set of K points, $\{z_i\}_{i=1}^K \in \Omega$, that correspond with $P(x)$.
3. Initialize the variable of weight; $J_i = 1$.
4. Pick a point $w \in \Omega$ according to $P(x)$.

5. Find the z_i closest to w ; denote its index by z_i^* .
6. Set $z_i = \frac{J_i * z_i^* + w}{J_i + 1}$ and $J_i = J_i + 1$.
7. Loop from Step 4 to Step 6 until the desired precision has been met.

3.1.1 Strengths and Weaknesses

Due to the nature of the K-means algorithm, the amount of machine computation required for the generating points to converge to the true Voronoi centroids is nearly constant throughout all dimensions. Once a weighted random number generator is developed from the density function, the computer needs only to create random points and average point positions to create the centroidal Voronoi tessellation. Averaging requires a low level of computation, and there is no significant difference to the processor between averaging one coordinate and four coordinates. The number of iterations required for convergence is quite large because of the algorithm's dependence upon randomly selected points.

The strength of K-means in higher dimensions lies in its use of random points to induce convergence rather than regional centroids. The computer can easily spit out random points, but calculating centroids, as Lloyd's Method does, involves numerical integration which is significantly more processor intensive. Despite the sheer number of iterations needed to create a centroidal Voronoi tessellation, K-means is often the quickest algorithm in higher dimensions.

The ease of one dimensional integration is the downfall of K-means. The number of iterations saved by referring to the positions of regional centroids instead of random points easily justifies spending a little extra processing time to integrate.

One must be wary of the fact that coding the necessary weighted random number generator is generally very difficult and time consuming, as is discussed in the following section.

3.1.2 Implementing the Density Function

K-means requires that points be selected randomly in accordance with a case-specific probability distribution, a task that can be implemented efficiently for discrete frequency distributions but only with difficulty for continuous ones. We are faced with the inverse problem of defining a random function to fit the specified density.

Implementation on One Dimension

First, the smooth case, on which we focused this research. The two dimensional density function can be handled as an extension of a one dimensional function, so we'll analyze that first. To create a *weighted random point* function on the interval $[0,1]$, begin by considering a *density function*, which is an integrable function $\rho(x)$ such that the probability of choosing a weighted random point in the interval $[a, b]$ is $\int_a^b \rho(x) dx$. Call this value $\rho([a, b])$, using ρ as both the density at a point and accumulated density (mass) over a subinterval of $[0,1]$. In order to randomly choose points which fit the density requirement ρ given an unbiased random number generator (such as those available on most computer systems) $rand : \emptyset \rightarrow [0, 1]$, we must introduce a

diffeomorphism $\phi : [0, 1] \rightarrow [0, 1]$, so that $(\phi \circ rand())$ gives output with the desired distribution.

Theorem 1 *The function $WRP = (\phi \circ rand) : \emptyset \rightarrow [0, 1]$, where $rand$ generates truly random output in $[0, 1]$ and $\phi^{-1}(x) = \int_0^x \rho(t)dt$, is a weighted random point function with characteristic density function ρ .*

Proof. The probability $P(WRP() \in [a, b]) = P(rand() \in \phi^{-1}([a, b]))$. Since ϕ is defined as the integral of a non-negative function, $\phi(t)$ is monotonic and therefore invertible almost everywhere. (That is, $\phi^{-1}(x)$ can be defined for almost every $x \in [0, 1]$, with discontinuities at values of x where $\phi(t) = x$ over finite intervals of t .) Since ϕ is monotonic, $\phi^{-1}([a, b]) = [\phi^{-1}(a), \phi^{-1}(b)] = [\int_0^a \rho(t)dt, \int_0^b \rho(t)dt]$. By virtue of its true randomness, $P(rand() \in [t_1, t_2]) = t_2 - t_1$, so $P(rand() \in \phi^{-1}([a, b])) = \int_0^a \rho(t)dt - \int_0^b \rho(t)dt = \int_a^b \rho(t)dt$, and hence WRP has the desired probability distribution.

Implementation in Higher Dimensions

To extend the one dimensional case to two dimensions (and beyond), we consider the probability function as function $\rho_X(x) = \int_0^1 \rho(x, y)dy$, the probability density in x , and $\rho_{x_0}(y) = \rho(x_0, y)$, the probability of density in y given a fixed parameter x_0 . Random values of x and y may then be computed as detailed above.

Note that unless WRP can be determined analytically, generating random points according to the distribution ρ is a computationally intensive feat involving the inverse of an integral. However, if we note that ρ is the derivative of WRP , and consider that in practice, sometimes ρ is a contrived function that qualitatively approximates a phenomenon, the mathematical modelling process is well-enough served by defining WRP to fit the qualitative requirements and beginning from there.

Implementation of Discrete Density Functions

However, if we are considering a discrete universe, K-means becomes far more computationally feasible in any number of dimensions. First-off, note that it is trivial to interpret n -dimensional vectors in a finite space as numbers in a one dimensional set with and appropriately redefined distance function (If the maximum value in any dimension is k , simply reserve one field of 2^k bits for each of the n dimensions, and let the distance function operate on the appropriate bits.) Let M be an array of values culled from a phenomenon (such as colors from a digitized picture). To choose values randomly with frequency distribution equal to that of M , simply choose a position in M randomly and read its value. If one had the data only in value-frequency pairs, the computation detailed at the top of this section would be necessary to invert the data relationship. However, integration is replaced by finite sums, which are computationally simple to invert exactly.

3.2 Lloyd's Method

Given a region Ω and a density function $\rho(x)$,

1. Using a Monte Carlo method, chose some initial set of K points, $\{z_i\}_{i=1}^K \in \Omega$.

2. Construct the Voronoi tessellation $\{V_i\}_{i=1}^K$ of Ω associated with the points $\{z_i\}_{i=1}^K$.
3. Compute the mass centroids of the Voronoi regions $\{V_i\}_{i=1}^K$ found in Step 1; these centroids are the new set of points $\{z_i\}_{i=1}^K$.
4. Loop between steps 2 and 3 until the desired precision has been met.

3.2.1 Strengths and Weaknesses

Lloyd's method is a much more direct algorithm than K-means, and therefore the number of iterations needed to create the centroidal Voronoi tessellation is considerably less.

In higher dimensions, Lloyd's Method is generally slower than K-means due to the complicated process of numerical integration. However, Lloyd's Method allows the shape, size, and position of the Voronoi regions to be manipulated easily. By choosing a desirable density function, one can crunch the Voronoi regions in the center of the tessellation, induce rectangular Voronoi regions, or even cause them to crowd around several distinct points in the region. This will be further discussed in Section 4.

Lloyd's Method is extremely efficient in one dimension; the simplicity of one dimensional integration and the lack of necessary iterations allows it to be the better solution in this situation.

3.2.2 Tweaking the Computational Speed of Lloyd's Method

Guided by a suggestion from Dr. Gunzburger, we experimented with an enhancement of Lloyd's Method. He predicted that it would be possible to anticipate the movement of a centroid given its current and previous position, so he advised us to replace Step 3 in Section 3.2 with the following.

- Calculate the centroid c_i of V_i
- Set $z_i^{new} = 2c_i - z_i^{old}$

The new algorithm moves the generating points of the Voronoi region twice the distance of prescribed displacement of Lloyd's Method. Hypothetically, this accelerates convergence by allowing the generating points to travel larger distances per iteration toward the Voronoi centroids.

Performance in One Dimension

During the testing of the algorithm in one dimensional space, we found that it almost always decelerated the convergence. We concluded that the new algorithm frequently caused a generating point to overshoot its Voronoi centroid. Since Voronoi regions are created with respect to all generating points, this behavior creates a misalignment of all generating points. If the error is relatively small, the point can redirect itself and spend a few extra iterations to return to its Voronoi centroid without substantially harming the rest of the generating points. In this case, the time saved by increasing increments is usually cancelled by the time elapsed while the algorithm corrects itself; as a result, the efficiencies of the experimental algorithm and Lloyd's Method are nearly equal. But in most cases, the error is large enough to create region-wide chaos which causes at least one generating point to continually bounce back and forth over its Voronoi centroid.

We felt that if we lowered the ratio by which the distance was increased, there would be fewer instances of generating points skipping over their Voronoi centroids. We began a new experiment consisting of the algorithm

- Calculate the centroid c_i of V_i
- Choose a ratio r such that $1 \leq r \leq 2$
- Set $z_i^{new} = (r)c_i - (r - 1)z_i^{old}$

and comparing the performance of different values of r . Our initial generating points were randomly selected from the set $[0, 1]$ and were utilized to create a centroidal Voronoi tessellation on the region $[0, 1]$. Our data, as shown in Appendix A, suggests the optimal value of r is in the range $1.6 \leq r \leq 1.8$ and can boost performance by as much as forty percent.

Considering the relatively small space between the generating points of a centroidal Voronoi tessellation with many regions, our hypothesis was that the lesser values of r would perform best in such situations. By the same logic, we expected the greater values of r to strive when creating centroidal Voronoi tessellations with fewer regions. However, after examining the data, we concluded that the number of Voronoi regions had little to do with the performance of r ; it was affected primarily by the average distance of the initial generating points from their final resting places at the Voronoi centroids. We had hoped to derive a formula to calculate which ratio to use given some sort of initial information. Obviously, the positions of the Voronoi centroids are unknown until the completion of the algorithm, so producing a formula based on initial conditions is seemingly impossible.

Performance in Two Dimensions

The algorithm’s relatively strong performance in one dimension raised hopes that it would also accelerate Lloyd’s Method in two dimensional space. This would be extremely useful because, as described in Section 3.2.1, each eliminated iteration results in a significant reduction in processing time.

Although we did not have the resources to properly experiment with the algorithm in two dimensions, we included the tweaked version of Lloyd’s Method in some of our later simulations and observed that it offers little to no advantage over the original Lloyd’s Method. We attribute this finding to the inherent differences between one and two dimensional movement.

By the description of Lloyd’s Method, the generating points replace the centroids of their respective regions and, from the updated generating points, the new centroids of the Voronoi regions are calculated. In a one dimensional setting, the placement of the new centroids is bounded linearly; they can only be created on the same line which passes through the generating points. Thus, the direction of movement of the generating points is rather predictable. However, the creation of new centroids in two dimensional space is not linearly confined, and they often stray from the prescribed linear path of the generating points. This causes the anticipated movement of the generating points to be misdirected. If the deviation is small, a slight performance gain is recorded during the iteration, while there is a small decrease in performance if the deviation is large. Over the duration of the algorithm, these peaks and valleys combine to produce an overall performance very similar to the original Lloyd’s Method.

4 Qualities of density functions

Centroidal Voronoi tessellations minimize energy by anchoring Voronoi centers near local maxima of the density function (peaks). In this section, we will describe the qualitative shape of the density functions we used in our research. As shown in the source code in the Appendix, the general density function is a sum of primitive functions, and each primitive function has several user-definable parameters. Easily modifiable, the software allows functions from each of three families. The families are

Exponential $Ae^{-h_x(x-x_0)^{w_x}-h_y(y-y_0)^{w_y}}$

Sinusoidal $A[1 + \sin(\frac{2\pi}{f_x}(x - x_0))]^{n_x}[1 + \sin(\frac{2\pi}{f_y}(y - y_0))]^{n_y}$

Hyperbolic $Ax^{n_x}y^{n_y}$

The qualitative meaning of the parameters are:

- A varies a term's relative importance in the overall density function.
- x_0, y_0 places a peak at (x_0, y_0)
- $h_x (h_y)$ influences sharpness of peak in the $x (y)$ direction.
Larger values steepen the density gradient.
- $w_x (w_y)$ influences width of peak in $x (y)$ direction. Larger values increase the radius of the region in which density is substantial.
- $f_x (f_y)$ indicates the number of peaks along the horizontal (vertical) line containing (x_0, y_0)
Peaks are separated by a distance $\frac{2\pi}{f_x} \cdot (\frac{2\pi}{f_y})$
- $n_x (n_y)$ influences narrowness of the peak. Larger values decrease the radius of the region in which density is substantial.

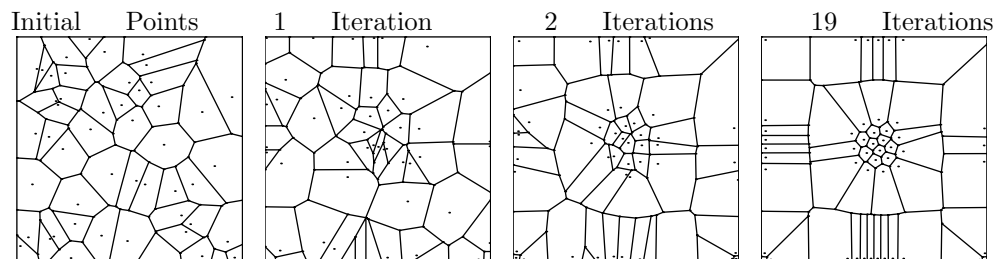
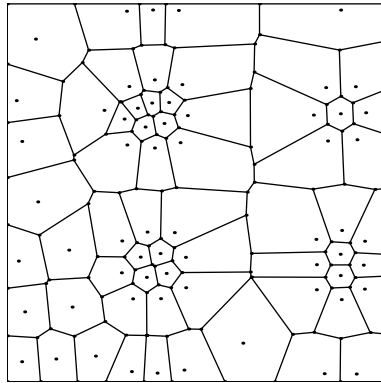
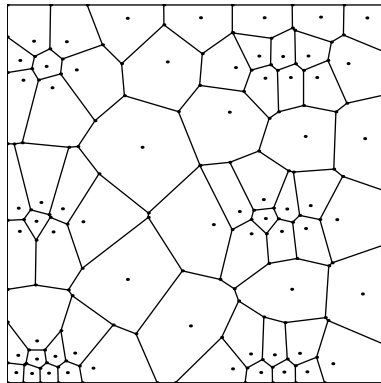


Figure 1: Voronoi diagrams of the function $\rho(x, Y) = (1 + \text{Sin}[3\pi x + 3])^6(1 + \text{Sin}[3\pi y + 3])^6$

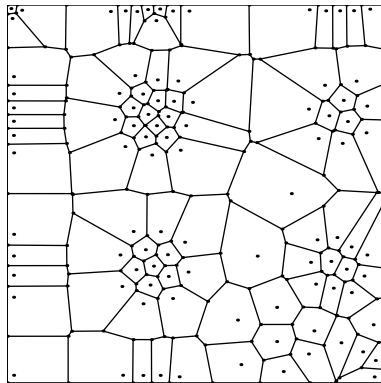
$$\rho(x, y) = e^{(-5(x-.7)-4(y-.2))} + 4((1+\sin(8\pi(x-3)))^4(1+\sin(10\pi(y-4)))^2) + 2((x-.9)^{-2.5}(y-7)^{-30})$$



$$\rho(x, y) = e^{(2x^3+4y^7)} + (\sin(3\pi x + 1))^4(\sin(5\pi y + 1))^6$$



$$\rho(x, y) = 9((1+\sin(8\pi(x-3)))^4(1+\sin(10\pi(y-4)))^2) + 2((x-.9)^{-2.5}(y-7)^{-30}) + 2(x^7(y-2)^9)$$



5 Suggestions for Further Research

- In Lloyd's method, one could record the energies of intermediate centroidal and Voronoi tessellations as they converge to a centroidal Voronoi tessellation. Computational rate-of-convergence data for different families of density functions may be enlightening.
- Search the real numbers for an optimal tweak value for Lloyd's method. Perhaps the optimal value has a recognizable dependence on the dimension, number of Voronoi regions, or density function.
- We conducted all of our research on the unit square. Investigating other shapes of regions such as polygons, circles, tori, polygons with holes, and infinite or semi-infinite regions may prove fruitful.
- Computational time and storage space limitations restricted our data collection to tessellations of less than 100 points each. The rate-determining step in Lloyd's method is the generation of Voronoi regions from a given set of Voronoi centers, so a faster algorithm could allow the exploration of asymptotic behavior as the number of points $\rightarrow \infty$.
- Explore the possibility of representing density with a matrix mapping instead of a density function. Find the advantages and disadvantages of such an algorithm.
- Creation of centroidal Voronoi tessellations in higher dimensions.

Special Thanks to Steve Fortune for the use of his Voronoi tessellation code.
www.voronoi.com for providing links to Voronoi code.
Dr. Gunzburger for solid suggestions and academic advice.

The programming codes written during our research may be downloaded at:

<http://www.hcs.harvard.edu/~roger/CVT/>

Appendix: Source code listings

File names are current as of February 19, 2001. Source code is available online at <http://www.hcs.harvard.edu/~roger/CVT/> , and the most important code is printed here.

program_name.pl is a self-executing perl script.

program_name.cc is C++ code.

program_name.mx is Mathematica code.

1. *tweaklloyd.pl* - Implements Lloyd's method in 2-dimensions by coordinating a collection of C++ programs.
 - (a) (*voronoi*) - Steve Fortune's program to generate a description of a Voronoi tessellation (in the form of a set of line segments and rays) for given set of centers. (Code not included here.)
 - (b) *2dlloyd_final.cc* - Inputs *voronoi* tessellation data output by Steve Fortune's *voronoi* program, corrects errors, converts from plane tessellation to unit-square tessellation, writes Mathematica *.mx* file that draws tessellation, writes *.lcvo* datafile readable by *newc.cc*
 - (c) *newc2.cc* - Inputs *.lcvo* file generated by *2dlloyd_final.cc*. Analyzes line segment data to find polygons and writes *poly* file listing a vertex list for each polygon. Calculates centroids of polygons using density function defined in the file *density.cc*, writes output to a *.vi* file.
 - *density.cc* - Defines the density function used by *newc2.cc*
 - *quad.cc* - Performs numerical integration on a triangle via 7-point quadrature.
 - *mylib.cc* - Secretarial function library for *newc2.cc*
 - *namemaker.pl* - Writes a *.mx* file to animate the graphics generated by a series of runs of *2dlloyd_final.cc* .
2. *Voronoi.pl* - Coordinate *K_Means.cc* and visualisation via *voronoi* and *2dlloyd_final.cc*
 - *K_Means.cc* - Performs *K_means* algorithm to find centroidal Voronoi tessellation, writes list of Voronoi centers to *.cvp* file
 - *mymath.h* - header file for constructs used by *K_Means.cc*
3. *1dlloyd_tweak.cc* - finds Voronoi centers for 1 dimensional centroidal Voronoi tessellations, using internally coded density function.
4. *random.pl* - generates random points in the unit square.
5. *strip.pl* - strips comments from *.cvp* file created by *K_Means.cc*, writing an input file suitable for Steve Fortune's *voronoi* program.
6. *flower.mx* - Sample *.mx* file generated by *2dlloyd_final.cc*