Approximating $\delta - covers$ for estimation of Hausdorff dimension using Genetic Techniques

Benjamin J Edwards
Mathematics and Computer Science Department
South Dakota School of Mines and Technology
Rapid City, SD 57701
Benjamin.Edwards@gold.sdsmt.edu

March 11, 2005

Abstract

Mandelbrot defined a set as fractal if it has a non-integer Hausdorff dimension. Unfortunately in the case of these fractal sets, Hausdorff dimension is often extremely difficult to calculate analytically or numerically. A pivotal element in estimating and Hausdorff measure and dimension is a $\delta - cover$ of a given set. Genetic techniques are used to create optimal $\delta - covers$ for any given set. The $\delta - cover$ consists of a collection of disks with a given center and radius. Because this definition of a population member moves beyond the traditional binary representation presented by Holland [3] novel techniques for mutation and recombination are developed. Mutation occurs by adding individual disks to the candidate $\delta - cover$. The fitness of each population member is determined through a linear combination of different covering possibilities. An example run is shown and future work discussed.

Benjamin James Edwards
Mathematics and Computer Science
South Dakota School of Mines and Technology
Rapid City, SD 57701
Benjamin.Edwards@gold.sdsmt.edu

1 Introduction to Fractals and Hausdorff Dimension

The applications of fractal geometry to physical systems has been pursued since the introduction of fractal sets by Mandelbrot in 1982 [1]. Mandelbrot defined a fractal set as simply a set with non-integer Hausdorff dimension. Hausdorff dimension is preferable to other dimensional quantifiers as it is defined for all sets and because it is based on measure it is easy to manipulate mathematically [2]. Unfortunately it is difficult to determine analytically or through numerical computation.

To begin a review of Hausdorff dimension we define U to be a non-empty subset of $\mathbb{R}^n$, and $|U|$, or the diameter of U, to be the greatest distance between any two points x, $y \, \epsilon U$. If $\{U_i\}$ is a countable collection of sets of diameter less than δ such that $F \subset \bigcup_{i=1}^{\infty} U_i$ with the diameter of U being less than δ for all i, we say $\{U_i\}$ is a $\delta - cover$ of F [2].

Suppose that $F \subset \mathbb{R}^n$ and $s \, \epsilon \, \mathbb{R}$ and $s > 0$. For any $\delta > 0$ Hausdorff Measure can be defined as

$$\mathbb{H}^s_\delta(F) = inf\{\sum_{i=1}^{\infty} |U_i|^s\}$$

where $\{U_i\}$ is a $\delta - cover$ of F. More simply we are examining all the covers of F with a diameter of at most δ. We want to minimize this sum of diameters raised to the sth power. If we allow δ to decrease the class of permissible covers of F is reduced. So the infinum of $\mathbb{H}^s_\delta(F)$ increases, and we approach a limit as $\delta \to 0$. We write the s dimensional Hausdorff measure of F as

$$\mathbb{H}^s(F) = \lim_{\delta \to 0} \mathbb{H}^s_\delta(F)$$

.
This limit exists for any subset F of $\mathbb{R}^n$, though the limiting value can be 0 or ∞, and the proof of which is beyond the scope of this paper. In fact this value is 0 or ∞ for most values of s and it is the point at which the limit changes from one to the other that we are interested in. We define

$$D_{\mathbb{H}}(F) = sup(s : \mathbb{H}^s(F) = \infty) = inf(s : \mathbb{H}^s(F) = 0)$$

as the Hausdorff dimension of a set F [2].

It is clear to see a pivotal portion of the estimation of Hausdorff dimension is the minimal $\delta - cover$. We can approach this as an optimization problem, and therefore apply genetic techniques to develop the minimum $\delta - cover$ given a set to be studied. The development of a minimal $\delta - cover$ is analogous to the sphere packing problem in $\mathbb{R}^2$. This paper concerns itself with creation of this cover using a genetic algorithm to develop a soft disk packing requirement.

2 Disk Packing

2.1 History and Background

The problem of disk, or circle packing, has been well studied and has long history. The problem was first studied by Kepler in 1611, when he examined the sphere packing problem and attempted to prove that the densest possible is the cubic or hexagonal packing, and that its packing density is $\eta_{Kepler} = \frac{\pi}{3\sqrt{2}}$. This became known as the Kepler Conjecture the proof of which was elusive for many years. The approximation for the maximum density was refined many times over, before the density was finally proven by Hales in a series of papers culmination in 1998 [4].

In 2 dimensions the problem is simplified, and an equally great amount of research has been done. For disks of equal size the maximal packing density is given by $\eta_h = \frac{1}{6}\pi\sqrt{3}$ and was proven by Gauss to be the hexagonal arrangement seen in Figure 1. This packing density is defined for disks of equal size given a specific boundary.

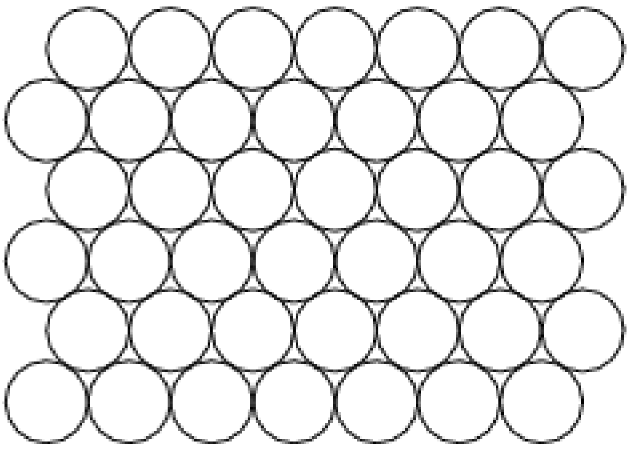

Figure 1: Hexagonal Packing

Alternatively disk packing with disks of various sizes is also heavily studied under the topic of discrete conformal mappings. This is an area of complex analysis. A conformal mapping is a transformation $w = f(z)$ that preserves local angles. Any analytic function is conformal at any point where it has a nonzero derivative. It should be noted that when applied to disk packing, all the disks remain tangential and within a given boundary [4].

2.2 Soft Disk Packing

The problem we wish to study is disks with a maximal radius, so we can create a $\delta - cover$. Moreover because the desired end result is an approximation of a cover of a given area, we can allow a “soft-cover” to be possible. That is, a cover which contains some overlap as well as boundary violations. We will use our soft disk packing method to develop a $\delta - cover$ which can be used in the investigation of Hausdorff dimension in later research.

3 GA Work

3.1 Problem Space

To examine this disk packing problem we need a way to examine an area that needs to be covered. A simple approach is taken. Each area to be covered is represented as a black and white image. The area to be covered is represented in black and any extraneous area is represented in white.

Each disk is represented by a colored disk drawn centered within our region. In this way we can differentiate between the area to be covered, the disks themselves and any area outside that which we are trying to cover. Moreover we can define a maximum size for each disk and therefore define a specific $\delta - cover$.

3.2 Population Members

To approach this problem with a genetic technique a novel method of representing the population members had to be determined. Rather than define each population member in a traditional binary string format, each population member consists of a list of center points and radii for each disk. A linked list is used as this allowed for easy implementation using the C++ standard template library, and allows each population member to be of dynamic size, allowing for smaller or larger collections of discs depending on the area being investigated.

Though the base of the population member is the list of disks of various sizes, more information about the population member was stored in a class comprising the list as well as other values. These values included total area covered, overlapping area, appropriately covered area, and extraneously covered area. These values were critical in calculating the fitness of each population member, and were stored within each population member’s class for computational efficiency.

To begin with an examination of the population member, we note that given any area we are trying to fill with disks several occurrences can happen. Firstly, our disk may fall completely within an area that needs to be filled. This case can be seen in Figure 2.
Secondly, a disk can fall in an area that need not be filled. Finally, a combination of the two can happen, a disk can fall on the border of one of these regions. The second and third cases can be seen in Figure 3 and Figure 4 respectively.

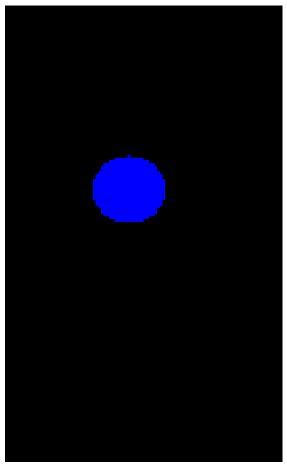

Figure 2: A covering disk

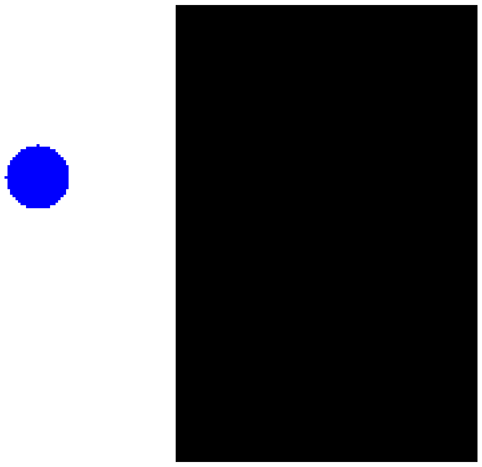

Figure 3: A non-covering disk

In addition to these three cases a variation can happen on any of them in which two disks in close proximity overlap each other. This can be seen in Figure 5.
In the disk packing problem we strive to ensure zero overlap as well as a maximal number of disks covering appropriate area, and a minimum number of disks outside the area to be covered. In this way we can begin to create a cost function and rank our members.

We approach this as a minimization problem. Adding cost to a population member based on negative properties, and ranking them according to minimum cost. To determine how a particular population member stands up, we simply draw each disk in its appropriate place on the provided image. Then we can check pixel by pixel with the original and classify it. We draw each disk in a color different from the black and white covering in the original so we can distinguish where each disk lies and what areas it occupies. We can classify the pixels in one of two ways: first it can be a non covered pixel which should be covered,

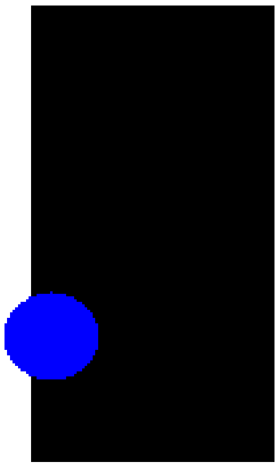

Figure 4: A half covering disk

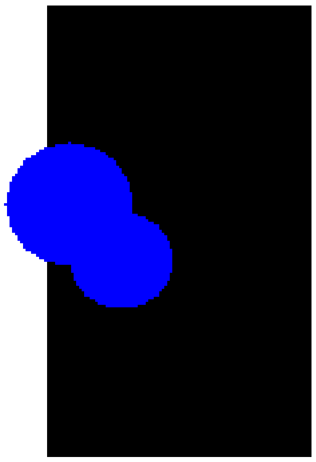

Figure 5: Overlapping disks

or it can be a covered pixel which should not be. We do this by simply matching each to a specific value. If on the image which we have drawn our disks there is a pixel which is colored but on the original is white, we can count that pixel as an extraneous pixel P_E. If the pixel is black on both images we can surmise it is a non-covered or missed pixel, P_M. We outlined before that we also wanted to reduce overlap in our determination of the $\delta - cover$. To do this we can simply calculate the sum of the areas of the disks comprising a particular population member, and take the difference between this and the actual number of colored pixels present. This estimate of overlap will be denoted P_O.

One more factor that deserves attention in our calculation of cost is the number of disks we are using to cover the area. It is beneficial to encourage a large number of disks in the cover, as it increases the probability of smaller disks occupying space in between larger disks. We can now construct our cost function. The cost function can be seen below.

$$Cost = \lambda_0 P_M + \lambda_1 P_E + \lambda_2 P_O - \lambda_3 S$$

Where each λ represents a constant and S is the size of the cover. Different λ values will yield better results for different areas to be covered, and may need to be altered depending on the area presented.

3.3 Creating the Population

With our population members defined we can create an initial population. The size of the each of the members of the initial population is kept constant. A disk is created with a random location within the image. Then a random radius is assigned to the disk. With the random disk created we simply add it to the population member as a piece of genetic material and repeat the process for the constant size defined for population members. This process is repeated once again until the desired initial population size is reached.

3.4 Crossover

Because our population members are not of a conventional binary nature a new approach had to be taken when creating new offspring. However the list nature at the core of our population members provides an obvious solution. Because the location of the disks is randomly distributed throughout the list we can be assured that taking genetic material from any portion of the list is as effective as taking genetic material from any other portion of the list. So the makeup of each offspring of two parents was determined in the following way. Firstly the size of the list that makes up the majority of the offspring is determined by an average of the two parents. Before creation this size is stochastically altered so as to provide population members with varying amounts of genetic material.

After the size of the population member is determined its genetic makeup must be determined. This is done by generating a random number between 0 and the sum of the two ranks of the two parents. If the number is larger than the lower ranked, or better, parent a disk is taken from the top of that parent and added to the child. If the number is less than the rank of the higher ranked parent a disk from the bottom of the higher ranked parent and added to the child. This process is repeated till the number of disks in the child is equal to the aforementioned genetic material size.

In this way a larger portion of the lower ranked parent is present in the child and 'better' genetic material is maintained in offspring.

3.5 Mutation

Mutation is important to any genetic solution to a problem. To move our population members out of possible local minima a simple approach was taken. Random disks were added to the population members while creating children. This was achieved by adding a certain number of disks based on the number of disks already in the population member. This number is a random value between 0 and a percentage of the genetic material present in the population member. This percentage, which we will refer to as the mutation rate, was altered in attempts to achieve maximal convergence. High mutation rates yielded the best results and seemed to prevent the algorithm from stagnating in local minimums.

4 A Basic Run

A simple run of the program began as follows. An ellipsoid region embedded in a white space is selected as the region needed to be filled. This region can be seen in the Figure 6.

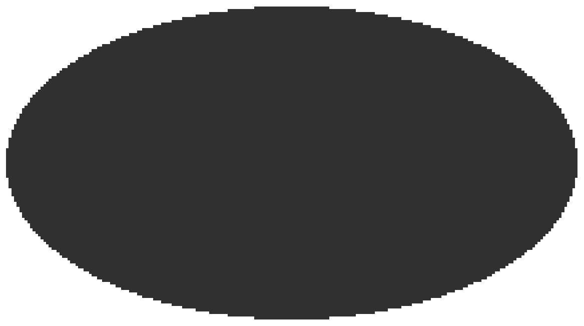

Figure 6: A sample area

An initial population is created based on selected parameters. For this particular region, we will start with an initial population of 500 members, with each population member having 200 disks within it.

The initial population is then sorted according to the following cost function. These values were selected using a great deal of trial and error and seem to yield the best results.

$$Cost = P_M + 2.1P_E + 2P_O - 7.8S$$

After the population is sorted, recombination takes place in the manner outlined above. After creation each child is mutated at a given rate. For this particular run a mutation rate of 10% was used. Each child is added to the population and the population is once again sorted.

At this point we only wish to maintain a certain number of particularly fit individuals. This is done in a dynamic manner. First an upper and lower bound are set. The population is not allowed to grow beyond the initial population size, but is also not allowed to sink below half of the population size. The exact number of population members is left to the fitness of each member. If the population becomes large we wish to have more stringent criteria for life within the population, but for low population numbers we wish to maintain a higher number of individuals. This is done by taking a variable range of members within a certain percentage of the most fit member. The following equation demonstrates how this is implemented within the code.

$$Cost < BestCost(1 + \frac{IntialPopulationSize}{CurrentPopulationSize}c_0)$$

Where c_0 is a constant. This simply means if a population member's cost is within a certain percentage, with a minimum of c_0 it is allowed to stay in the population. If it is above this cost value it is removed. For our run we will select $c_0 = .175$.

In addition to this method of trimming the population, it was found that the algorithm would stagnate on certain values when a dominate population member was created. Because a great deal of the most fit member's genetic material is dispersed among the population, the population becomes homogeneous and any optimization is slowed to a stop. To avoid this each population member was given a lifespan of between 20 and 30 epochs. In this way a population member would not dominate and create a homogeneous population.

This technique was used on our ellipsoid region and was run for 1000 epochs. Figure 7 is a plot of the cost as a function of epochs. It should be noted that convergence slows down considerably towards the end of the run. The result of the run can be seen in Figure 8.
It should be noted that this is a soft cover, and that disks remain completely outside of our region to be covered. Additionally, there is still some overlap as well as missed pixels within the region.

The main opponent of the outlined genetic program throughout its creation and subsequent testing was time complexity. Image sizes usually range in hundreds to thousands of pixels on a side, and during each epoch each pixel must be checked for hundreds of newly generated population members.

For this reason many quantifiers were calculated as the children were created to save computation cycles. For instance during the creation of each population member, whether child or initial member, after each disk was added to the population member the disk was drawn on the image for comparison. Unfortunately this restricted the ways in which creatures could be mutated, as disks could not be removed or changed, because it may affect any number of vital statistics present in the calculation of the cost function.

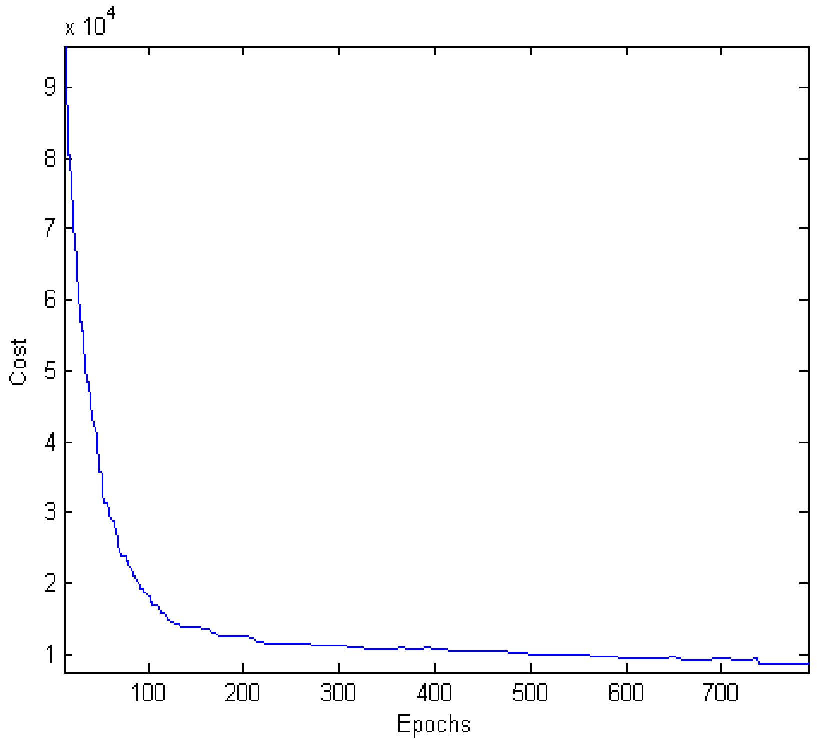

Figure 7: Cost vs. Epochs for Ellipsoid region

5 Conclusion

The application of genetic techniques to the problem of disk packing presents an interesting problem. While many techniques can be used for well defined boundary areas, random, complex, or poorly defined boundary areas may require a more general approach, such as the one presented in this paper. The use of genetic algorithms makes the complexity of solution method greatly reduced, but also results in imperfect solutions. Fortunately for estimation purposes, a certain about of leniency is allowed.

The use of a set of disks to define a population member may be extensible to other areas, where binary or tree representations of population members may not be appropriate. As with many genetic techniques, this one was highly sensitive to a myriad of different parameters within the structure of the algorithm, including the population size, the size of population members, the computation of the cost of the members, and the rate of mutation. For any given area to be analyzed specific tuning of the parameters may be necessary.

The results of other runs were similar to the example above. While these covers are certainly far from perfect they may suit the need for computation of Hausdorff dimension. This soft packing method gives us a fair estimation of what a $\delta - cover$ may look like for a given fractal. Further work will need to be completed to determine whether the $\delta - cover$s created by the algorithm are viable for fractal dimension calculation.

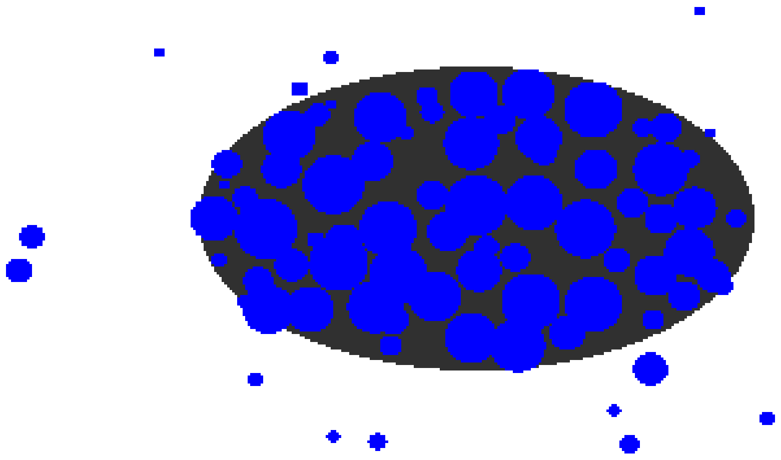

Figure 8: Covered Ellipsoid region

References

[1] Benoit B. Bandelbrot, *The fractal geometry of nature*, W. H. Freeman, 1982.

[2] Kenneth Falconer, *Fractal geometry: Mathematical foundations and applications*, John Wiley and Sons, West Sussex, England, 1990.

[3] John H. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, Anna Arbor, MI, 1975.

[4] Eric W. Weisstein, *Circle packing*, http://mathworld.wolfram.com/CirclePacking.html.