**Multi-Dimensional Hill Climbing**

**By Walter Gress**

**Optimization of a 4-Dimensional Time Dependent Analysis Data Set using optimization of four parameters and multidimensional hill climbing algorithm.**

This algorithm is divided into several stages, at which different operations are performed using the given algorithms.

*Stage 1*

A four dimensional surface t = f(x,y,z) is to be determined via sampling of runs of the algorithm, each sample returning a time associated with its x, y, and z parameters.

To begin the first stage, a minimum value of 0 is used for all three independent variables (x,y,z). A median must be determined where the time trade off between a single variable and the time variable t is optimal. This value is used as a maximum for the variable. This time trade off is determined empirically through repeated runs of the algorithm using fixed values for all of the variables except the test variable, which is incremented by steps of size k, where k is a hard-coded constant.

Examining Trends and finding the best time/data tradeoff:

For analysis of each parameter, we do not want to exceed a time value that exceeds polynomial time, that is T(n) = O(nk) for some constant value k. This means if the plot of time vs. the parameter we are analyzing exceeds O(n2) time (where n is the parameter and T(n) is the time gathered), we will call that the cut off point for that parameter, considering O(n2) is a reasonable runtime and comparable with quick sort, shell sort, and other sorting algorithms, while not much larger is associated with NP Hard and NP Complete problems.

Since we do not want to exceed the function n2 in the steepness of our curve, the tractability test can be performed by taking the slope of n2, which is the derivative 2n at every point n. We then compare that to the data given, calculating the slope from the parameters current time minus its last time, divided by the difference of the parameters current value minus the parameters last value. If the time begins to exceed n^2, we cease our incrementing of it and continue on with the other parameters. (Assuming that they do cease, ) Once all parameters cease, the loop ends and the tractability endpoints are found for the variables.

*Stage 2 (incomplete: how to determine a surface from point data?)*

It is necessary to find the optimal values for all three independent variables to the dependent variable t interacting together, not simply on their own, otherwise Stage 1 would be sufficient. Stage 2 involves generating a hypersurface (4D surface in this case) with boundaries determined by the minimum value (0) and median values (determined in stage 1) of the data set.

It is necessary to generate a hypersurface using known, discrete algorithms allowing the data to be processed like a surface on a computer (later, the forward difference is used to take the partial derivatives of such hypersurface).

The procedure that follows is for a typical, three-dimensional surface.

The gathered data points are constructed as follows:

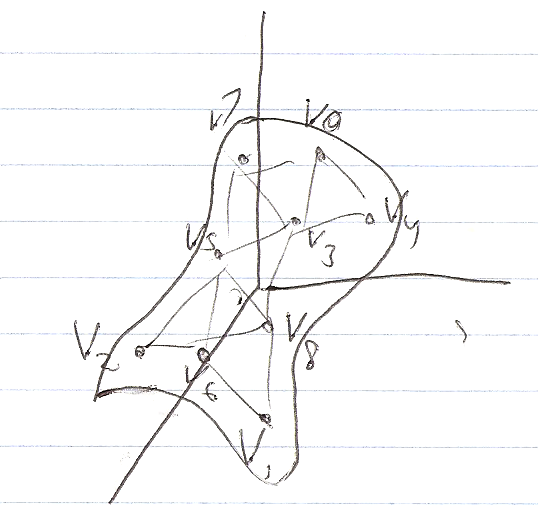
S = {V0, V1, V2, V3, V4, V5, …., VN}

V0 = {X0, Y0, Z0 }

Vi = {Xi, Yi, Zi }

It is necessary to construct a polygonal surface out of our data points listed above.

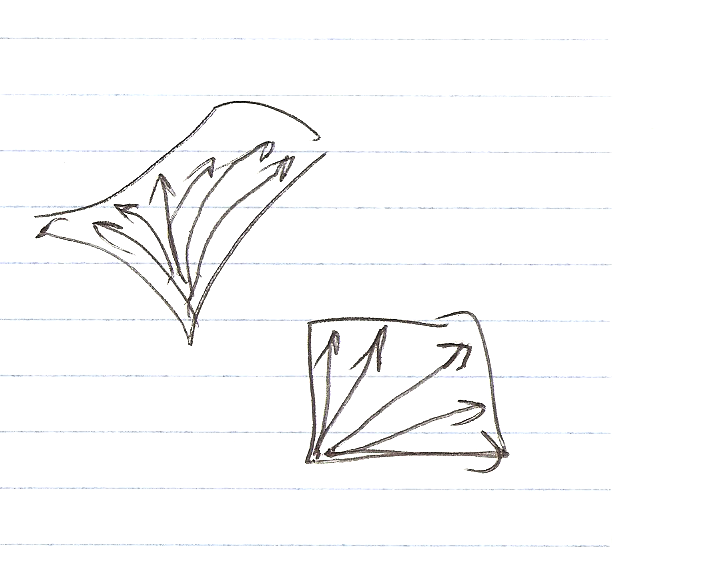
We would like to construct something similar to:



Because the data points (hereafter called vertices) are arbitrarily placed in the list and unsorted, it is necessary to determine their location relative to their neighboring vertices. By starting with the minimum vertex ( min[x], min[y], min[z]) we can find the farthest most corner of the surface. From there we can calculate the distance from each vertex in the surface to the corner vertex just identified.

Now we would like to generate a planar, polygonal surface using planes of three points (the minimum required for a plane). Any three points form a plane, and any plane plus a non-collinear point also forms a plane. Using the corner vertex of the surface, we can use the distance data gathered to find the closest two vertices to the corner vertex. From this we can generate the first plane, and then using the distance data, go to the next closest vertex to the plane and as long as it is non-collinear, build a plane from that using the new vertex and the previous plane.

The result is something similar to below:



With the polygonal surface construted, we can proceed to the next stage, stage 3, and calculate the finite partial derivatives via forward difference operation on the polygonal surface model.

*Stage 3 Determining partial derivatives of surface (incomplete, how to determine a partial derivative programmatically)*

The surface can be represented in continuous form as t = f(x,y,z), where t is a function of three independent variables, x, y and z.

The four dimensional surface model allows for a four dimensional surface representing the combination of the partial derivatives of f(x,y,z) to be formed, whose low point is representative of the most efficient values for (x,y,z). The partial derivative carries over the idea of the derivative to higher dimensions ([Wikipedia](http://en.wikipedia.org/wiki/Multivariable_calculus#Fundamental_theorem_of_calculus_in_multiple_dimensions))

For example, in a function of one variable, y = f(x), the first derivative is a function representing the slope of f(x).Where the slope is 0, an inflection point occurs, indicating either a maximum or minimum in the original function. This theory carries over to three dimensions, z = f(x,y), where the derivative again represents the shape of the three dimensional surface in a similar fashion, f’(x,y) -> 0 representing points of little slope in the original three dimensional shape.

Given the surface as a series of discrete values an application of finite differences in more than one variable can be applied to get a discrete version of the partial derivative. Taken from Wikipedia (en.wikipedia.org/wiki/Finite\_difference)

For F(x,y,z) the limit definition would be as follows (y and z are held constant):

therefore making the partial finite differences:

And again, the partial finite differences are analogous to a discrete form of the partial derivatives.

*Stage 4 Minimizing surface (possibly with hill climbing algorithm) to determine locus for optimal efficiency*

Three partial derivatives are taken with respect to the generated surface. (Possibly using discrete forward difference operations)

The partial derivative with respect to x (y and z held constant), the partial derivative with respect to y (x and z held constant), and the partial derivative with respect to z (x and y held constant).

Values are entered into the three partial derivative equations, stepwise by a constant k, from the minimum (0) to the median data point for the variable calculated in stage 1.

When the partial derivatives result in or approach 0, the values for its respective variables are recorded and stored as a 4-dimensional “inflection” point or low point in the original 4-d generated surface. A simple hill climbing algorithm can be used to determine 0s or approximate 0s on the surface using the hill climbing algorithm along with the partial derivatives.

The recorded values should be the resulting optimum values and should have their respective t values compared against each other to determine the overall optimum of the algorithm run. They are considered the optimum quantities for three variables since at the point on the derivative there t value is 0, indicating that there is an inflection point in the original surface. At inflection points there is always either a (local) max or minimum. An arbitrary test can determine whether it is the former or latter.