ROCSCIENCE INC.

Locating General Failure Surfaces in Slope Analysis via Cuckoo Search

Aleck Wu

May - Aug, 2012

SUMMARY

Geoslope stability analysis is an important area in geotechnical engineering. Proper analysis of a slope geometry can lead to safer development as well as a better understanding of the site in question. The main objective of slope stability analysis is to locate the critical failure surface—a surface along which the rock mass or soil is most likely to fail. This "likelihood-to-fail" is quantified by the factor of safety (ratio of total shear strength go shear stress) associated with each unique surface. For any generic failure surface described by a polyline of N points, the factor of safety can be described as a function $F(\boldsymbol{P})$, where $\boldsymbol{P} = [t_0, (x_1, y_1), \dots, (x_i, y_i), \dots, (x_{n-2}, y_{n-2}), t_{n-1}]^T$; a parametric value along the slope surface is used to denote the single freedom the entry and exit point has. One can locate the critical failure surface by minimizing $F(\boldsymbol{P})$. Due to geometric and kinematic constraints and conditions needed to satisfy a valid failure surface, $F(\boldsymbol{P})$ is highly discontinuous as well as often multimodal, rendering standard optimization techniques highly unreliable as well as inefficient.

Various global optimization techniques have been implemented over the years in locating the critical failure surface, including Simulated Annealing implemented in SLIDE—a 2D geoslope stability analysis software developed by Rocscience Inc. A recent population-based stochastic algorithm—Cuckoo Search—shows great promise, outperforming some more traditional global optimization algorithms such as Particle Swarm and Harmony Search under standard test functions—taking less function evaluations to achieve the same level of solution accuracy.

The current work implements an improved variant of Cuckoo Search, coupling with the Local Monte-Carlo (LMC) optimizer, in searching for the critical failure surface in SLIDE. Cuckoo Search incorporates a random walk and random solution generations to escape local minima, while LMC is a local explorer optimizing an existing failure surface through constantly varying the position of each polyline vertex. Various refinements were also made on top of the Improved Cuckoo Search specific to the problem at hand. The algorithm was implemented in C++.

The hybrid method proposed above—Improved Cuckoo Search with LMC—was found to be superior to Simulated Annealing in locating the true global minimum in slope geometries where more than one failure mode is present. The quality of solutions found by Improved Cuckoo Search hybrid with LCM is comparable to those found by Simulated Annealing, with a much improved computation time of on average three times faster.

TABLE OF CONTENTS

1 INTRODUCTION

Slope stability analysis is an important and essential step in the planning and building of various structures such as dams, embankments, etc. along the sides of slopes. Improper analysis can lead to damages to expensive structures and human life when soil masses fail and slides under load.

A typical slope failure is shown in Figure 1. Under load, the top soil mass will fail under shear, and dislocate from the rest of the slope. In two-dimensional slope analysis, this line of shear stress failure characterizing the location of the failure is called a failure surface (in 2D analysis the failure surface is considered to extend into the page). The stability of the soil mass atop the failure surface is quantified by a numerical factor of safety, given as the ratio of shear strength to shear stress under the specific load (i.e. weight of soil); a factor of safety under 1 indicates instability.

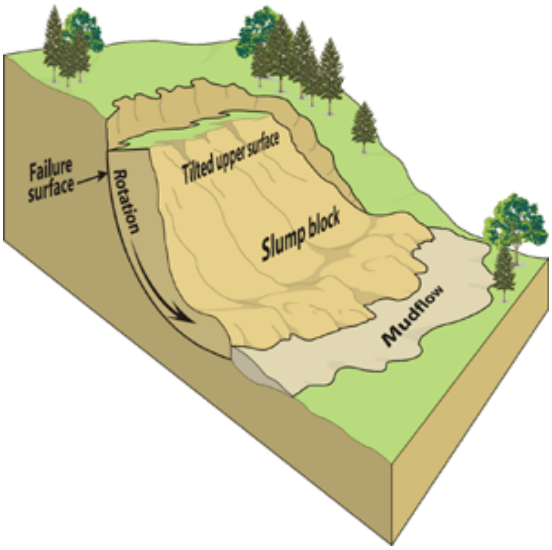

FIGURE 1 - TYPICAL FAILURE SURFACE

Prior to slope stability analysis, it is unclear as to whether or not a slope will fail, and if so, along which failure surface it will do so. Therefore, at the core of slope stability analysis is the location of the weakest failure surface—the surface along which the soil mass will most likely fail.

Given a failure surface, there are various different methods to calculate a numerical factor of safety. A well established class of methods is the Limit Equilibrium Methods (LEMs). These methods divide the slope geometry into slices, and through satisfying forces and/or moments equilibrium, converges on a numerical factor of safety. Some popular LEMs include the Ordinary Method of Slices, Bishop's Method, Janbu's Method, and Spencer's Method. For a more thorough description of LEMs, please refer to "Slope Stability Analysis and Stabilization—New Methods and Insights, Chapter 2, slope stability analysis methods" by Y.M. Cheng and C.K. Lau.

2 PROBLEM FORMULATION AND DESCRIPTION

2.1 DIMENSIONALITY

The problem of locating the critical two-dimensional failure surface is in essence a global optimization (minimization) problem. The function to be minimized is the factor of safety function, a function consuming a positional vector of the polyline describing a failure surface, and outputting a scalar factor of safety value associated with such a failure surface—$FOS = F(\boldsymbol{P})$, where

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{p_0} = slope(t_0) = (x_0, y_0) \\ \boldsymbol{p_1} = (x_1, y_1) \\ \dots \\ \boldsymbol{p_i} = (x_i, y_i) \\ \dots \\ \boldsymbol{p_{n-2}} = (x_{n-2}, y_{n-2}) \\ \boldsymbol{p_{n-1}} = slope(t_{n-1}) = (x_{n-1}, y_{n-1}) \end{bmatrix} \text{ is a solution vector,}$$

and $\boldsymbol{p_0}$ … $\boldsymbol{p_{n-1}}$ are 2D points defining a failure surface of n vertices.

The dimensionality of the problem is thus dictated by the number or vertices of the failure surface polyline; each inner vertex contributes to two extra dimensions, while single parametric values $t_0, t_{n-1} \in [0,1]$ are needed describe the entry and exit vertices of the failure surface since both must lie on the slope surface. Thus, a failure surface of n vertices (including entry and exit points) will translate to a $2n - 2$ dimension problem.

2.2 CONSTRAINTS AND DOMAINS

There are constraints which define the domains of each of the $2n-2$ control variables; these constraints are necessary in order to satisfy primary geometric and kinematic requirements of a potential failure surface.

The slope surface is first parameterized from left to right between 0 and 1, and as such,

$$\{t_0, t_{n-1} \mid t_0 < t_{n-1} \in [0,1]\}$$

define the entry and exit point of the failure surface. This generates points

$$\boldsymbol{p_0} = slope(t_0) = (x_0, y_0),$$

$$\boldsymbol{p_{n-1}} = slope(t_{n-1}) = (x_{n-1}, y_{n-1}).$$

2.2.1 BOUNDS FOR X-COORDINATES

In a failure surface of n-vertices including entry and exit points, $n-2$ equal-width slices are generated between x_0 and x_{n-1}; these slices bounds the set of x-coordinates such that

$$\left\{x_1 < x_2 < \ \ldots < \ x_{n-2} \mid x_i \in \left(x_{i-1}, x_{i-1} + \frac{(x_{n-1} - x_0)}{n-2}\right)\right\}.$$

The domain restrictions of the x-coordinates of the inner vertices are clear and logical; they arise by definition from meaning of the subscripts. In laymen's term, each inner x-coordinate must situate within its respective slice.

2.2.2 BOUNDS FOR Y-COORDINATES

Dynamic bounds for y-coordinates are used as proposed by Cheng (2007) such that a convex, kinematically feasible surface can result. The following section will first attempt to describe such bounds in words, followed by mathematical formulation.

1. Firstly, the entry and exit points are assumed to be already defined and valid on the slope surface. As well, the x-coordinates of the inner vertices must also be defined and validly bounded.
2. The y-coordinate of the first point immediately after the entry point is bounded by the slope geometry, namely the slope surface and bedrock at its corresponding x-coordinate.

3. For each subsequent point, aside from the geometric bound which applies as described in 2., a kinematic bound is imposed. A line is constructed from the 2 previous points. Another line is constructed from the previous point and the exit point. The y-values of these two lines at the point's respective x-value define this aforementioned kinematic bound.
4. The final bounding criteria for each subsequent point is then taken to be the interval contained by the highest of the lower bounds, and the lowest of the upper bounds.

Below is a more formal mathematical formulation.

1. Assume $t_0, t_{n-1}, x_1, \dots, x_{n-2}$ present and valid as described in previous section. Denote $y = S(x), y = R(x)$ the functions describing the slopeline and the bedrock respectively.
2. $y_1 \in \big(R(x_1), S(x_1)\big)$.
3. $y_i, i \in [2, n-1] \in \big(\max\big(R(x_i),\ lower_{i,kin}\big), \min\big(S(x_i),\ upper_{i,kin}\big)\big)$, where

$$lower_{i,kin} = y_{i-1} + \left(\frac{y_{i-1} - y_{i-2}}{x_{i-1} - x_{i-2}}\right)(x_i - x_{i-1}),$$

$$upper_{i,kin} = y_{i-1} + \left(\frac{y_{n-1} - y_{i-1}}{x_{n-1} - x_{i-1}}\right)(x_i - x_{i-1}).$$

As can be seen, the domains for the various control variables are highly dynamic and dependent on other control variables. Moreover, since a factor of safety for a failure surface is found iteratively, a numerical solution may not converge for some surfaces even though they are convex and appear to be kinematically feasible. As such, the function $FOS = F(\boldsymbol{P})$ is extremely discontinuous, and valid only for small "hyper-regions" in the $2n - 2$ dimensional hyperspace of the problem.

An illustrative example of dynamic bounds can be seen is Figure 2. X-domains can be generated at the same time, as depicted by the blue arrows, while y-domains must be generated from left to right, and the y-domain for the current vertex depends on the y-coordinate of the previous as well as right-most vertex.

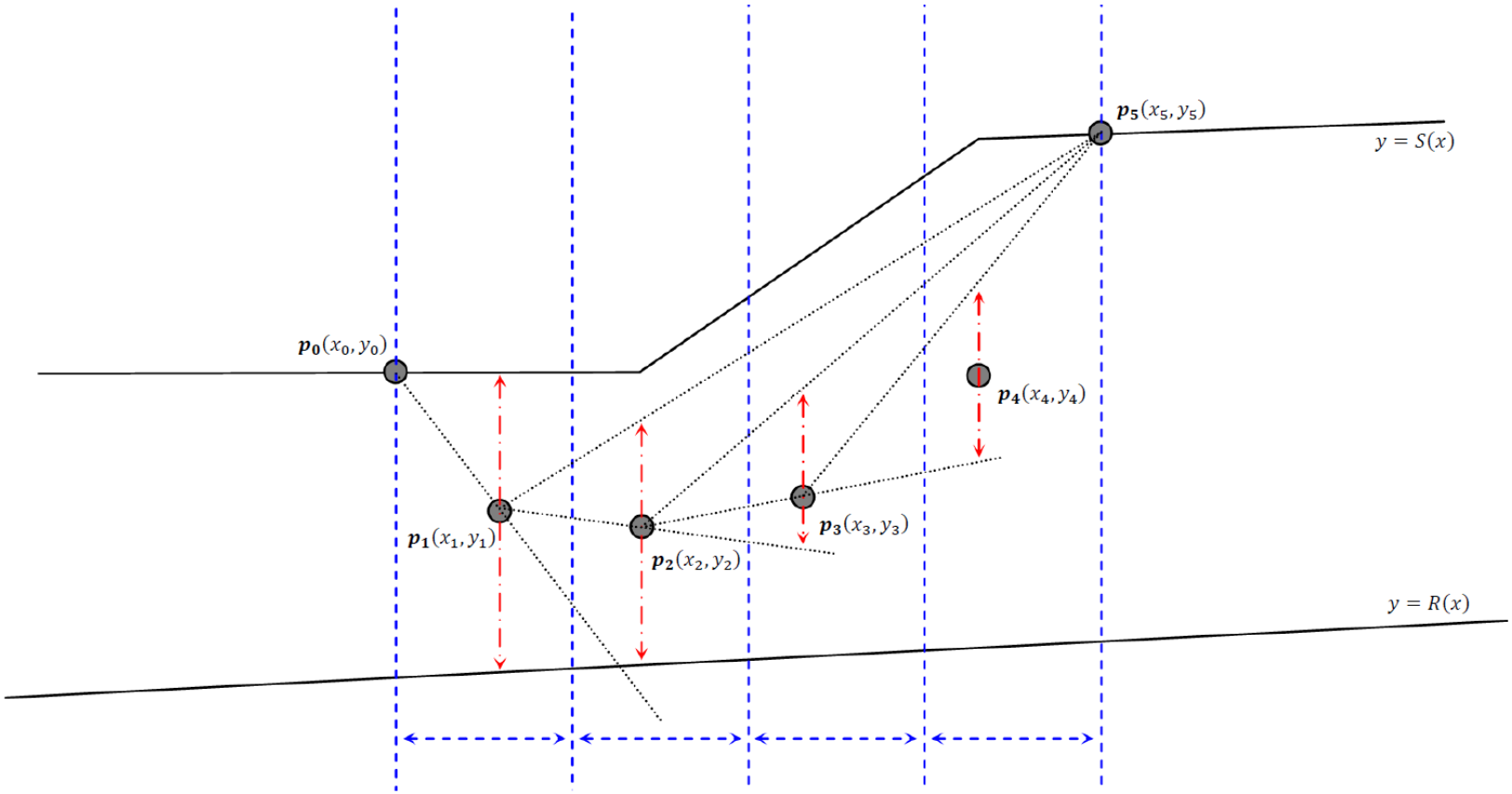

FIGURE 2 - DYNAMIC BOUNDS OF VERTICES

3 CUCKOO SEARCH

Cuckoo Search is a recent global optimization algorithm developed by Xin-She Yang and Suash Deb in 2009, inspired by the natural parasitic but successful behaviour of the Cuckoo species by laying their eggs in the nests of other host birds (of other species). In the short amount of time since inception, Cuckoo Search has been used in spring and welded beam designing, nurse scheduling, data fusion in wireless network sensors, etc., and obtaining better solutions than those which exists in literature.

3.1 INITIALIZATION

The CS algorithm starts by initializing a fixed number of N valid solutions vectors $\{\boldsymbol{P_0}, \dots, \boldsymbol{P_i}, \dots, \boldsymbol{P_{N-1}} \mid F(\boldsymbol{P_i}) > 0 \; exists \; \forall \; i \; \in [0, N-1]\}$. A fixed number of iteration I_{max} is also defined; generally, both N and I_{max} depends on the dimensionality of the problem. The solution vectors are sorted from worst fitness (i.e. highest factor of safety), to best fitness.

3.2 SOLUTION REFINEMENT

In each iteration in the original Cuckoo Search, for each solution vector $\boldsymbol{P_i}$, a temporary solution vector $\boldsymbol{P_{temp}}$ is generated by performing a random walk in the following fashion,

$$\boldsymbol{P}_{\boldsymbol{temp}} = \boldsymbol{P_i} + \alpha \oplus \boldsymbol{E_t},$$

where α is a problem related constant step-size, $\oplus$ denotes entry-wise multiplication, and $\boldsymbol{E_t}$ is a vector whose entries are taken from a probabilistic distribution. A random solution (i.e. $\boldsymbol{P_j}$) is selected and compared with $\boldsymbol{P_{temp}}$; $\boldsymbol{P_j}$ is replaced with $\boldsymbol{P_{temp}}$ if $\boldsymbol{P_{temp}}$ has better fitness. The solution vectors are then sorted from worst to best fitness again.

3.3 SOLUTION REJECTION AND REPLACEMENT

The last step in the standard Cuckoo Search algorithm is the rejection and replacement of a p_{rej} percent of the worst solutions with new randomly generated valid solution vectors. This ensures the global exploration ability of the algorithm never stops.

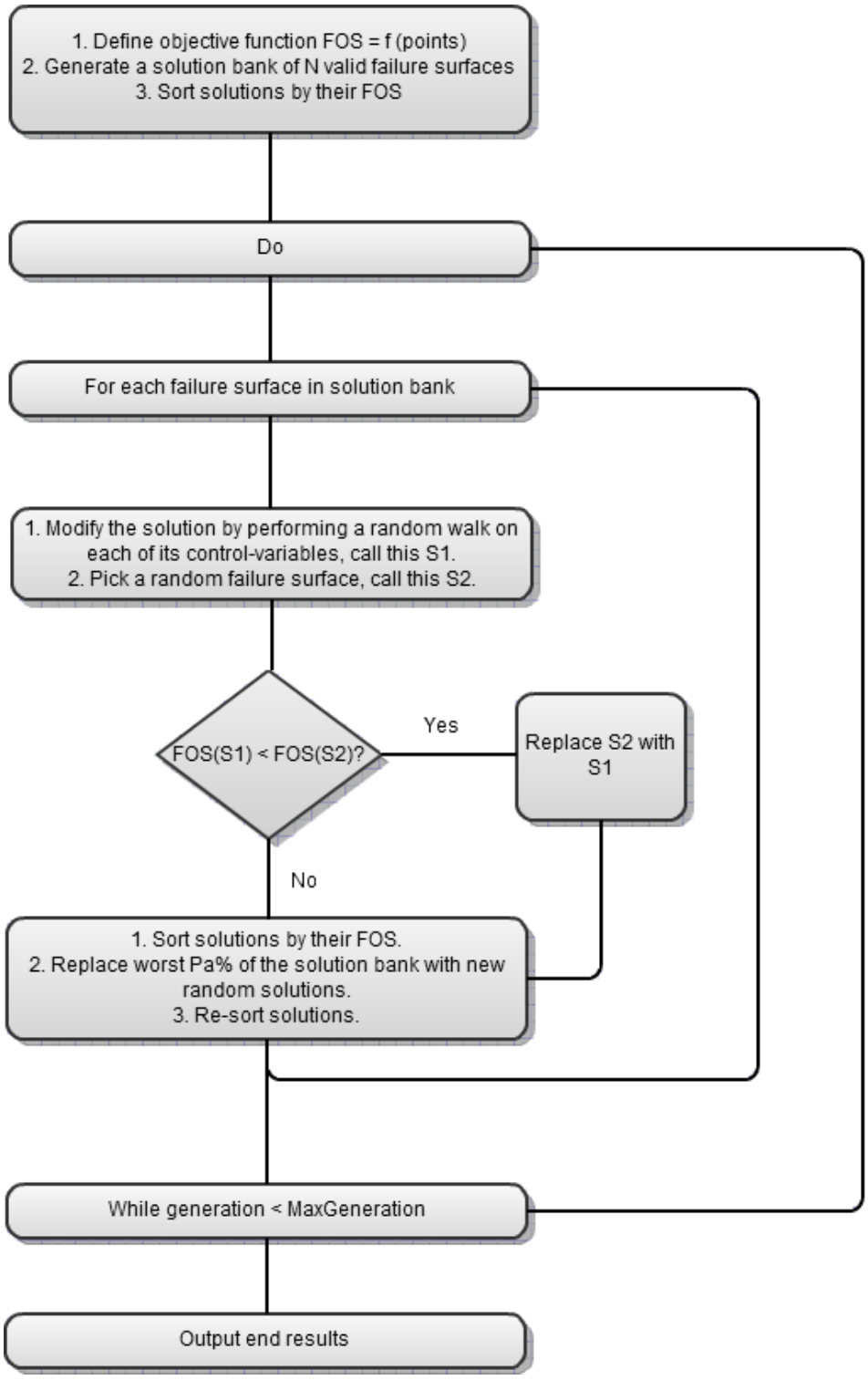

FIGURE 3 - FLOW CHART OF CUCKOO SEARCH

begin
Objective function $f(\mathbf{x})$, $\mathbf{x} = (x_1, ..., x_d)^T$
Generate initial population of
n host nests $\mathbf{x}_i$ $(i = 1, 2, ..., n)$
while *(t <MaxGeneration) or (stop criterion)*
Get a cuckoo randomly by Lévy flights
evaluate its quality/fitness F_i
Choose a nest among n (say, j) randomly
if $(F_i > F_j)$,
replace j by the new solution;
end
A fraction (p_a) *of worse nests*
are abandoned and new ones are built;
Keep the best solutions
(or nests with quality solutions);
Rank the solutions and find the current best
end while
Postprocess results and visualization
end

FIGURE 4 - PSEUDOCODE OF CUCKOO SEARCH

4 IMPROVED CUCKOO SEARCH

Various refinements have been made to the original Cuckoo Search algorithm. These modifications are mainly to increase convergence rate of the solution vectors.

4.1 VARIATION OF STEP-SIZE

For better global exploration abilities in earlier iterations and better local refinements in the latter stages, Improved Cuckoo Search proposed

$$\alpha(I_{current}) = (\alpha_{max}) \exp\left(\frac{\log\left(\frac{\alpha_{min}}{\alpha_{max}}\right) * I_{current}}{I_{max}}\right),$$

$$\boldsymbol{P}_{\boldsymbol{temp}} = \boldsymbol{P}_{\boldsymbol{i}} + \alpha(I_{current}) \oplus \boldsymbol{E}_{\boldsymbol{t}},$$

for which α_{max} and α_{min} are problem specific, $I_{current} \in [0, I_{max}]$ denotes the current generation.

4.2 VARIATION OF PERCENTAGE OF REJECTED SOLUTIONS

Again, to improve convergence rate, the number of solution vectors replaced by randomly generated solutions decrease over the iterations. In Improved Cuckoo Search, a formula for

$$p_{rej}(I_{current}) = p_{rej,max} - \left(\frac{I_{current}}{I_{max}}\right)(p_{rej,max} - p_{rej,min})$$

was proposed. It was also suggested that instead of the worst solutions being replaced, that each solution would have a p_{rej} percent chance of being replaced.

4.3 VARIATION OF REPLACEMENT OF SOLUTIONS

Instead of replacing rejected solutions with random solutions, it was proposed that the replacement solutions to be generated in the following way.

First, solution vectors are randomly shuffled, and two of such random permutations are stored inside integer arrays $perm1$ and $perm2$. For example, for a set of 5 reject solution vectors $\{\boldsymbol{P_0}, \dots, \boldsymbol{P_4}\}$, a possible permutation contained in $perm1$ might be $\{3, 2, 4, 0, 1\}$. For each rejected solution, the replacement solution will be generated by

$$\boldsymbol{P}_{i,repl} = \boldsymbol{rand}(\mathbf{0},\mathbf{1}) \oplus \left(\boldsymbol{P}_{\boldsymbol{perm1}[i]} - \boldsymbol{P}_{\boldsymbol{perm2}[i]}\right),$$

Where $\boldsymbol{rand}(\mathbf{0},\mathbf{1})$ is a column vector of length $2n-2$ (number of dimensions in the problem) filled with random numbers taken from a uniform distribution $U[0,1]$. To further speed up convergence, $perm2[i]$ can be replaced with $sorted[i]$ (by solution fitness) to put higher selection pressure on the solutions.

4.4 ON THE EFFECTIVENESS OF ASPECTS OF ICS IN LOCATING CRITICAL FAILURE SURFACE

It was found that all of the above techniques does allow a faster rate of convergence. However, due to the high dimensionality of the problem in question as well as the extremely discontinuous nature of the function, some of the Improved Cuckoo Search improvements hindered the global exploration ability of the algorithm in escaping local minima (i.e. secondary/tertiary failure modes of the slope). Hence only the variation of step-size modification was made to the original Cuckoo Search algorithm, as it was found that global exploration usually happens in the rejection-replacement phase of the algorithm.

5 REFINEMENT AND ADAPTATION OF CUCKOO SEARCH TO CURRENT PROBLEM

5.1 NO FREE LUNCH

In search and optimization computation, it is stated that any and all of such algorithms' performances are the exact same when averaged over all search and optimization problems. Performance of an algorithm can be described by its success to find the correct answer, as well as the amount of function evaluations it takes to achieve such a result. In many optimization problems, function evaluation is a computationally intensive process (i.e. iteratively converging a factor of safety given a failure surface), and the number of function calls dictate the speed of the algorithm.

As such, each algorithm must be tailored towards the specific problem it was intended to solve; a series of refinements specific and applicable only to the problem of critical failure surface searching has been implemented, and are detailed below.

5.2 GENERATION OF RANDOM SOLUTION VECTORS

Random solution vectors must be generated in the initialization phase as well as the rejection-replacement phase in each iteration. Since only a small and specific subset of the $2n-2$ dimensional space of the problem will return a valid factor of safety, it is important to ensure a high "success-rate" when generating such solution vectors.

Aside from the procedure described in Section 2.2 on generating valid control variables within each of their own domain, a further angular restriction is imposed; by default, the inside angle of each vertices of the failure surface is to be greater than 120 degrees (this of course, can be changed by the user).

Failure surfaces entering and exiting at the same elevation were also omitted. This can also be changed by the user in the case of a possible failure due to vertical external loading on a horizontal surface.

Instead of a uniformly distributed probability $t_0, t_{n-1} = U[0,1]$, each distinct segment of the slope surface (irrelevant of length) will have an equal probability of being selected as the segment for which the entry or exit point will lie. After such a slope segment is selected, uniform random distribution dictates the location of the entry and exit point along that segment. It was found that because different segments of the slope surface usually means a change in either slope geometry, or material properties, this modification allowed the algorithm to better survey the whole model.

Lastly, again due to the highly discontinuous nature of the control variable domains associated with this problem, simple checks were made to ensure that the failure surfaces generated do not cross and exit the slope surface near (but not at) the entry and exit points.

5.3 VARIATIONS OF SOLUTIONS

In the original Cuckoo Search algorithm, the variation of solutions phase follows

$$\boldsymbol{P}_{\boldsymbol{temp}} = \boldsymbol{P}_{\boldsymbol{i}} + \alpha \oplus \boldsymbol{E}_{\boldsymbol{t}}.$$

However, due to the high valid-domain dependence of the control variables, certain refinements were made to ensure a high success rate of $\boldsymbol{P}_{\boldsymbol{temp}}$ being a valid solution vector.

Firstly, the entry and exit points are varied parametrically along the slope surface. Then, all points are first "re-scaled" according to the new entry and exit points in the following manner:

1. Let $\boldsymbol{P}_{\boldsymbol{0,old}} = (x_{0,old}, y_{0,old}), \boldsymbol{P}_{\boldsymbol{n-1,old}} = (x_{n-1,old}, y_{n-1,old})$ be the pre-adjusted entry and exit points respective; and $\boldsymbol{P}_{\boldsymbol{0,new}} = (x_{0,new}, y_{0,new}), \boldsymbol{P}_{\boldsymbol{n-1,new}} = (x_{n-1,new}, y_{n-1,new})$ be the post-adjusted entry and exit points.
2. For each points $\{\boldsymbol{P}_{\boldsymbol{1}} = (x_1, y_1), \dots, \boldsymbol{P}_{\boldsymbol{i}} = (x_i, y_i), \dots, \boldsymbol{P}_{\boldsymbol{n-2}} = (x_{n-2}, y_{n-2})\}$

$$x_{i,new} = x_{0,new} + \left(\frac{x_{i,old} - x_{0,old}}{x_{n-1,old} - x_{0,old}}\right)(x_{n-1,new} - x_{0,new})$$

$$y_{i,new} = y_{0,new} + \left(\frac{y_{i,old} - y_{0,old}}{y_{n-1,old} - y_{0,old}}\right)(y_{n-1,new} - y_{0,new})$$

Secondly, a non-constant

$$\alpha(I_{current}) = (\alpha_{max}) \exp\left(\frac{\log\left(\frac{\alpha_{min}}{\alpha_{max}}\right) * I_{current}}{I_{max}}\right)$$

as was proposed by Improved Cuckoo Search was used, with $\alpha_{max} = 0.5$ and $\alpha_{min} = 0.05$.

Moreover, another step-size parameter $\boldsymbol{S}$ associated with each control variable is introduced—it is related to the domain of each control variable in association with what has been adjusted thus far.

1. $S_0 = S_{n-1} = t_{n-1,old} - t_{0,old}$ for entry and exit points, where the parameterized value is to be adjusted.
2. $S_i, i \in [1, n-2] = \frac{1}{4}\left(\frac{x_{n-1,new} - x_{0,new}}{n-2}\right)$. In other words, step-size for the middle vertices is a quarter of the width of each slice with regards to the already-adjusted entry and exit points.
3. Finally, the solution vectors are adjusted by following

$$\boldsymbol{P}_{\boldsymbol{temp}} = \boldsymbol{P}_{\boldsymbol{i}} + \alpha(I_{current})\boldsymbol{S} \oplus \boldsymbol{E}_{\boldsymbol{t}}$$

Lastly, every adjusted solution $\boldsymbol{P}_{\boldsymbol{temp}}$ is bound checked according to the geometric and kinematic requirement outlined in Section 2.2, and adjusted to the boundary extrema if out-of-bounds.

6 VERY FAST SIMULATED ANNEALING, LOCAL MONTE CARLOS

For a description and formulation of the very fast simulated annealing (VFSA) applied to this problem, as well as the Local Monte Carlos (LCM) optimizer already implemented in SLIDE, which compliments VFSA as well as Cuckoo Search, please see "Global Optimization of General Failure Surfaces in Slope Analysis by Hybrid Simulated Annealing, 2—Methods" by Su, Xiao.

7 RESULTS

The final Cuckoo Search algorithm tailored towards critical failure surface searching was executed on a total of 48 verification models and 19 customer models. These models include a variety of slope and embankment designs as well as multi-layered geometries, and will test the performance of the algorithm to find the global minimum in a wide variety of cases. In all cases, Spencer's method was used due to it being a limit equilibrium method satisfying both force and moment equilibrium.

The verification cases are published examples from engineering journals and conference proceedings, including a set of five slope cases as part of a survey sponsored by ACADS (Association for Computer Aided Design). The user-submitted cases consists of challenging slope geometries and extreme layering examples, submitted by SLIDE users. These often includes multiple modes of failure, and presents a true test to any global optimization algorithms.

Extensive testing of 30 runs of each file was conducted for both Cuckoo Search and Simulated Annealing. The Cuckoo Search with the necessary modifications and refinements described above along with the LCM performed very promisingly against Simulated Annealing. For all test files, Cuckoo Search found similar surfaces or completely different surfaces giving a much lower factor of safety as Simulated Annealing—at a computational time of on average 3 times faster. The results are summarized in Appendix I - Summary for averages of each model.

Is it also possible to display selected (or all) surfaces explored by the algorithm. This can result in a "gradient-like" coloring pinpointing on the critical failure surface, as well as display secondary or tertiary failure modes (local minima).

7.1 ON THE EFFECTIVENESS OF THE ALGORITHM

Both Cuckoo Search and Simulated Annealing are global optimization algorithms; thus, it is important to discuss their respective success in escaping from local minima to arrive at the global minimum.

The ability of Cuckoo Search to find the true global minimum is better than that of Simulated Annealing. This can be seen in a number of customer cases, where various local minima (secondary failure modes of a slope) exist. These files include User4, User4-revised, User6, User9, User13, and User14.

7.1.1 USER4 AND USER4-REVISED

These two are files with essentially the same slope geometry and material compositions, with User4-revised having a section of tension cracks along the top of the slope. Both of these cases have a point-like surficial failure mode with a factor of safety of 0.287, a point-like ledge failure with a factor of safety of 0.168 (global minimum), and a more substantial into-the-slope failure surface with a factor of safety of approximately 0.99. The two point-like failure modes can be seen clearly in Figure 5 by the patch of red dots indicating the center of the circle approximating the failure surface, while the failure mode with an FOS of 0.99 can be seen on the slope itself. The results of the 30 computational runs on the two files is summarized in Table 1 below.

	Surficial (FOS = 0.287)	In-Slope (FOS = 0.99)	Global Minimum (FOS = 0.168)
User4			
Cuckoo Search	8	0	22
Simulated Annealing	0	29	1
User4-revised			
Cuckoo Search	4	0	26
Simulated Annealing	2	27	3

TABLE 1 - USER4 RESULTS

As is evident, Simulated Annealing is not effective in escaping local minima in this situation. It is also important to note that although Cuckoo Search failed to find the global minimum a small fraction of the time, when the surfaces are displayed, it is clear that minima exist in locations of interests (including the area where the true global minimum lies)—although local exploration around that area during those few runs were not sufficient to retain those solutions as a candidate for the global minimum. Upon displaying the results in the SLIDE interpreter, users can easily further explore the area around which local minima were found.

In the above 2 customer files, the ability of Cuckoo Search to locate a near point-wise global minimum in the search space should be noted.

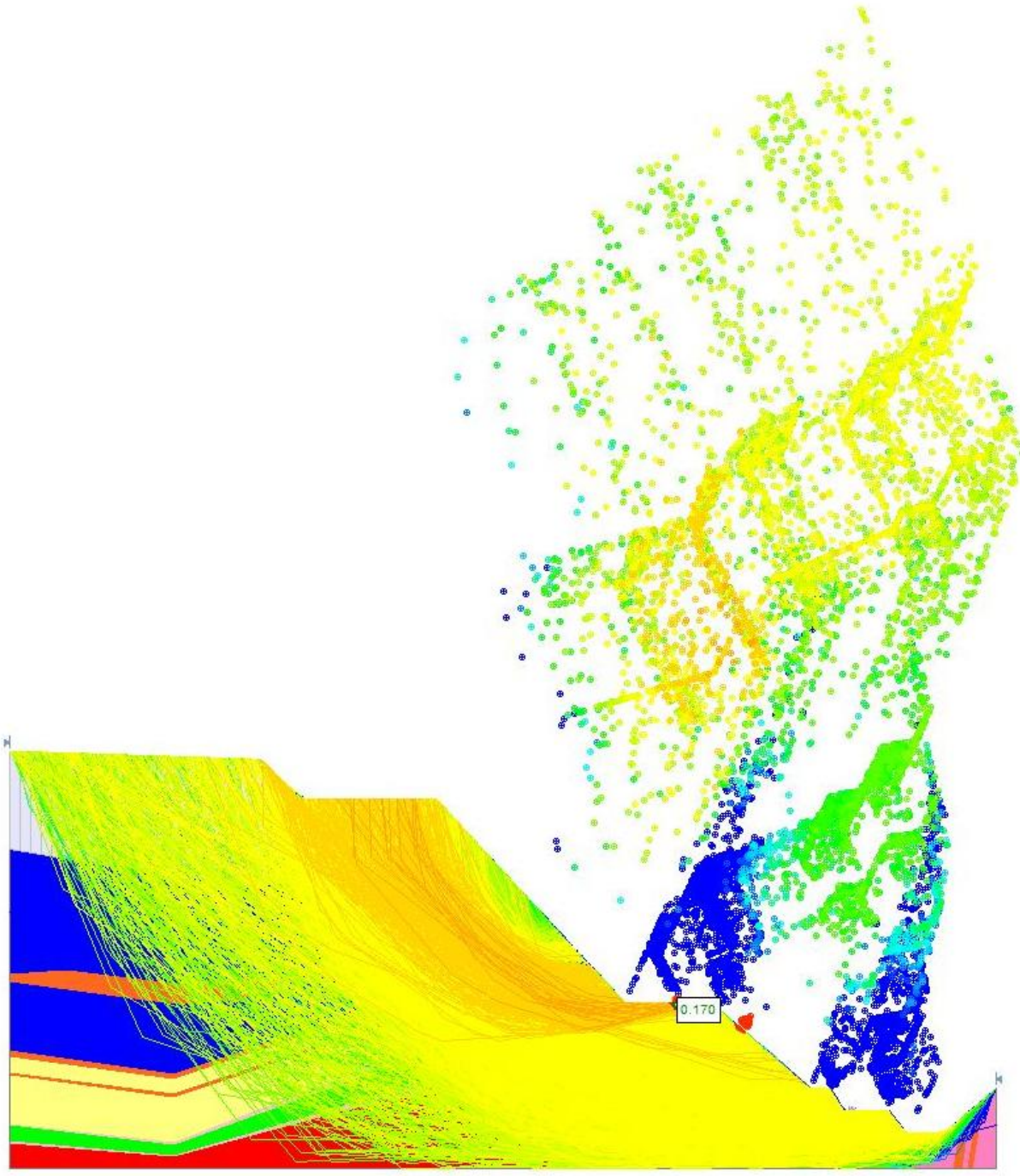

FIGURE 5 - USER4, CUCKOO SEARCH RESULTS

7.1.2 USER6, USER9, USER13, USER14

In all of the above files, there exist two failure modes which are geometrically far apart from each other (i.e. not in the same "valley" in the search space). With the exception of User6, for which a tertiary failure mode exists (however, neither Cuckoo Search nor Simulated Annealing have once mistakenly identified it as the global minimum, and is therefore insignificant in the current discussion). The results are summarized in Table 2 below.

User files 6, 9, 13, 14, with multiple failure modes (8 vertices)

User6	Local Minimum (FOS = 1.32)	Global Minimum (FOS = 1.23)
Cuckoo Search	0	30
Simulated Annealing	22	8
User9	Local Minimum (FOS = 1.23)	Global Minimum (FOS = 0.88)
Cuckoo Search	5	25
Simulated Annealing	29	1
User13	Local Minimum (FOS = 1.12)	Global Minimum (FOS = 1.82)
Cuckoo Search	2	28
Simulated Annealing	3	27
User14	Local Minimum (FOS = 0.93)	Global Minimum (FOS = 0.87)
Cuckoo Search	4	26
Simulated Annealing	5*	24

*a minima of FOS = 1.34 was found once

TABLE 2 - MULTIPLE FAILURE MODES CUCKOO SEARCH, SIMULATED ANNEALING COMPARISON

Again, in the more complicated customer cases with more elaborate soil layers and properties, Cuckoo Search was shown to be more—at times far more—superior to Simulated Annealing in escaping local minima and locating the true global minimum.

The files in question are displayed in Figure 6, Figure 7, Figure 8, and Figure 9. The different modes of failure can clearly be seen from the dense orange patches.

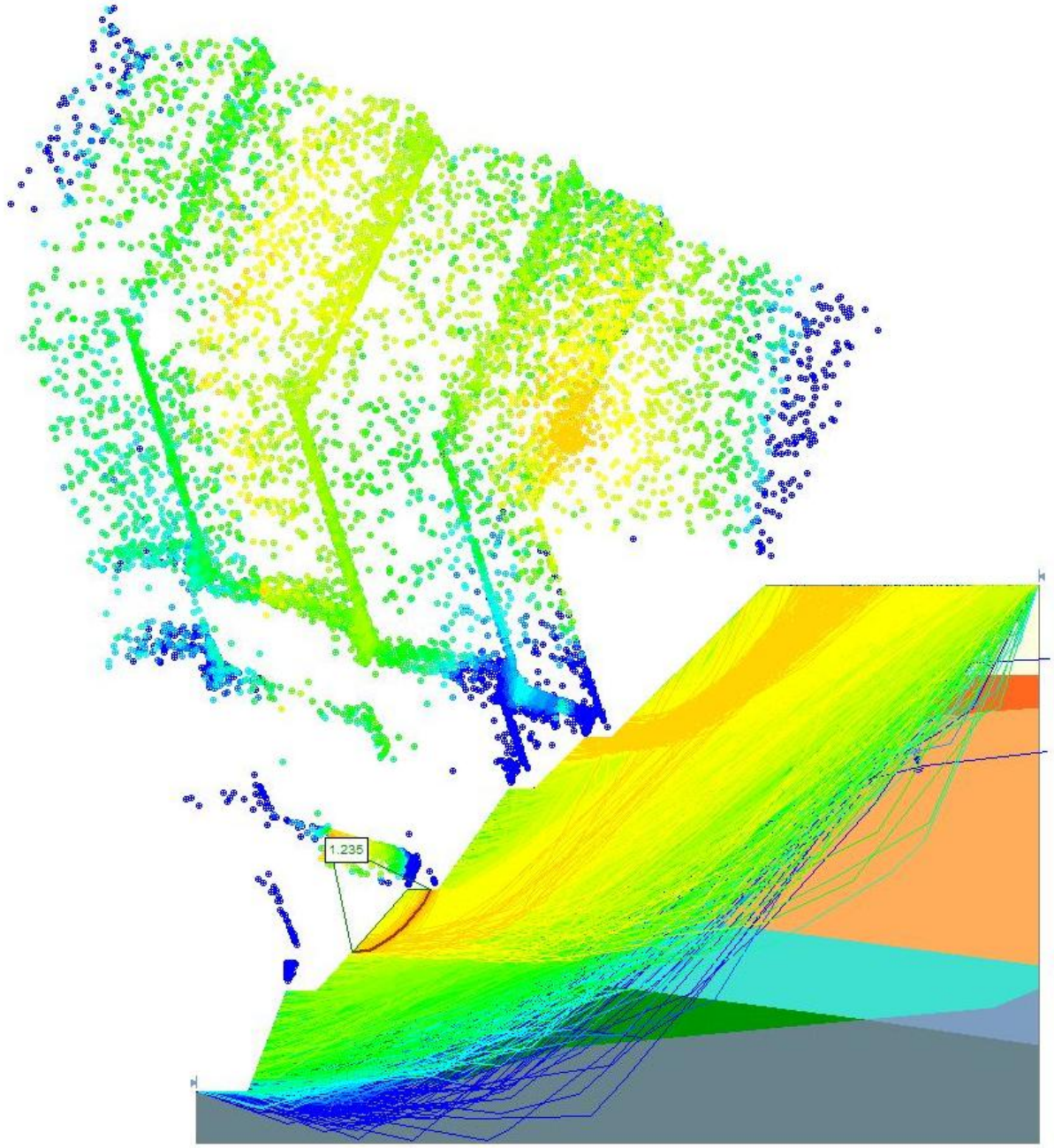

FIGURE 6 - MULTIPLE FAILURE MODES, USER6

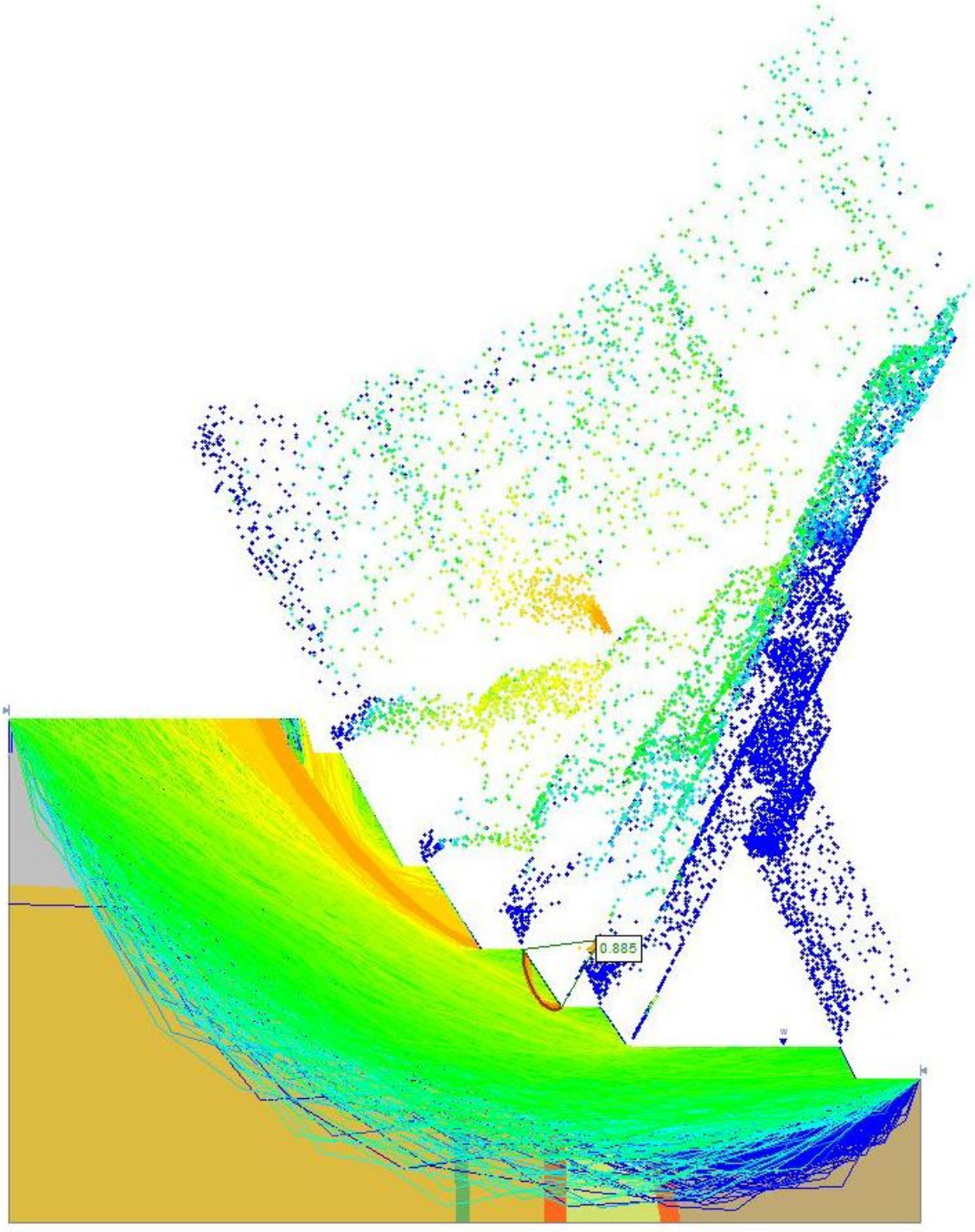

FIGURE 7 - MULTIPLE FAILURE MODES, USER9

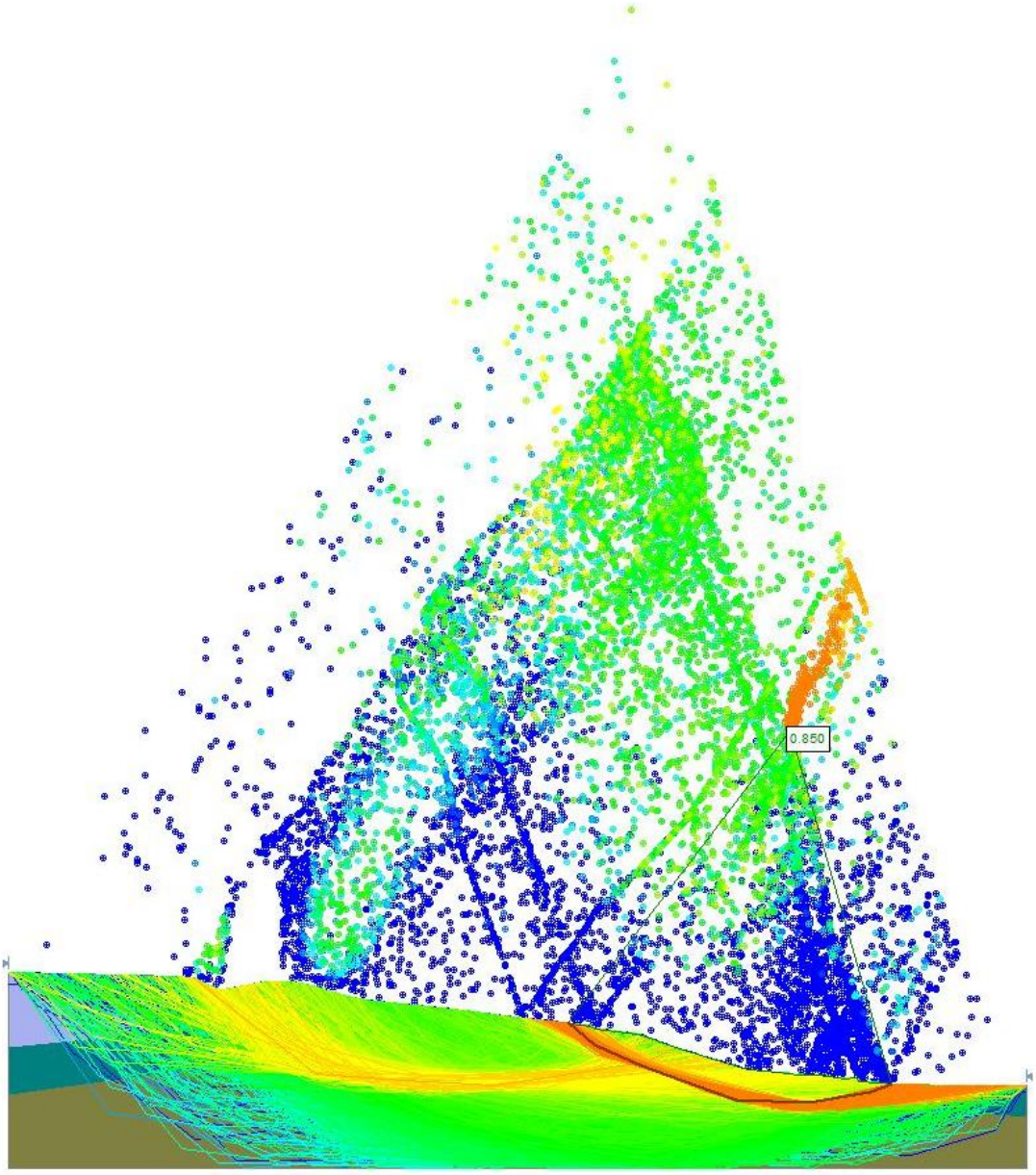

FIGURE 8 - MULTIPLE FAILURE MODES, USER13

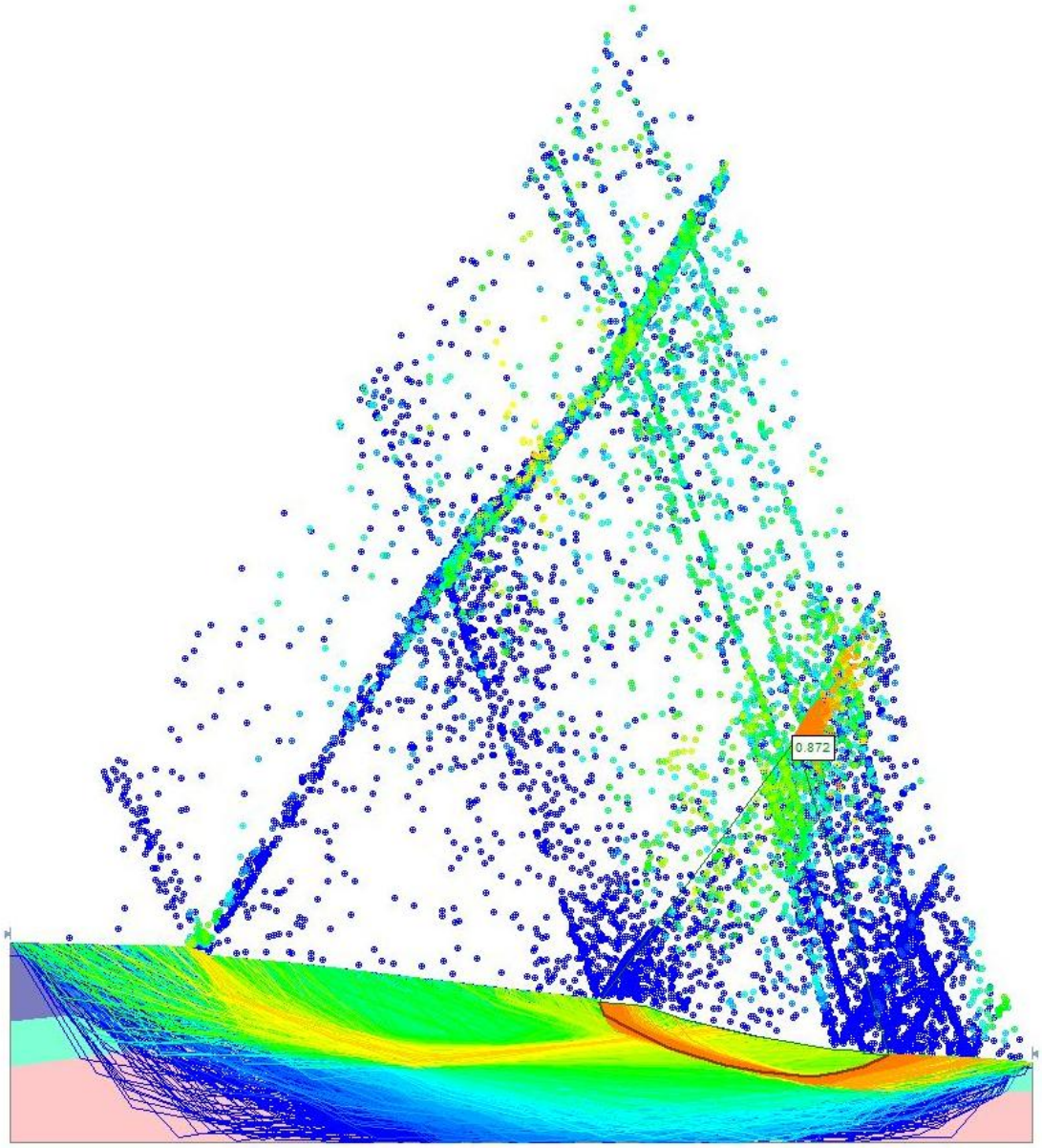

FIGURE 9 - MULTIPLE FAILURE MODES, USER14

7.2 ON THE EFFICIENCY OF THE ALGORITHM

For failure surfaces defined by 8 vertices (for a total dimensionality of 14), it was found that a solution bank size of 50 coupled with 500 iterations of the algorithm and a solution-rejection percentage of 40% is enough. This translates to

$$\left(50\frac{sol'n\ mod.\ \&\ eval.}{iteration}+(50*0.4)\frac{rand.sol'n\ gen.\ \&\ eval.}{iteration}\right)(500\ iterations)=35000\ func.eval.$$

This of course is overly optimistic and assumes a 100% success rate during random surface generations. As was mentioned, not all surfaces which satisfy the geometric and kinematic constraints will converge and evaluate to a valid factor of safety. For surfaces which do not, the surface is discarded and a new one is generated. This continues until a valid surface is generated.

The success rate of random failure surface generation depends highly on the actual slope model and its various properties; a success rate of 50% to 100% can be expected with most models. Assuming a success rate of 50%, 45000 function evaluations (factor of safety calculations) can be expected.

Even at 50% success rate, the number of factor of safety calculations are still small compared to Simulated Annealing. Because the bottleneck of the algorithm is the number of factor of safety calculations, Cuckoo Search sees a huge improvement in terms of computational speed over Simulated Annealing.

When averaged over all test files computed, Cuckoo Search performed three times (3x) faster than Simulated Annealing.

The percent difference of computational time between Cuckoo Search and Simulated Annealing is displayed in Figure 10, and is calculated by

$$\%\ difference=\frac{Simulated\ Annealing\ Time-Cuckoo\ Search\ Time}{Simulated\ Annealing\ Time}.$$

One can then calculate the amount of times Cuckoo Search is faster with the following formula:

$$Times\ Faster=\frac{Simulated\ Annealing\ Time}{Cuckoo\ Search\ Time}=\frac{1}{1-\%\ difference},$$

for example, 50% faster equates to a computational time half that of simulated annealing; 75% being four times faster, etc.

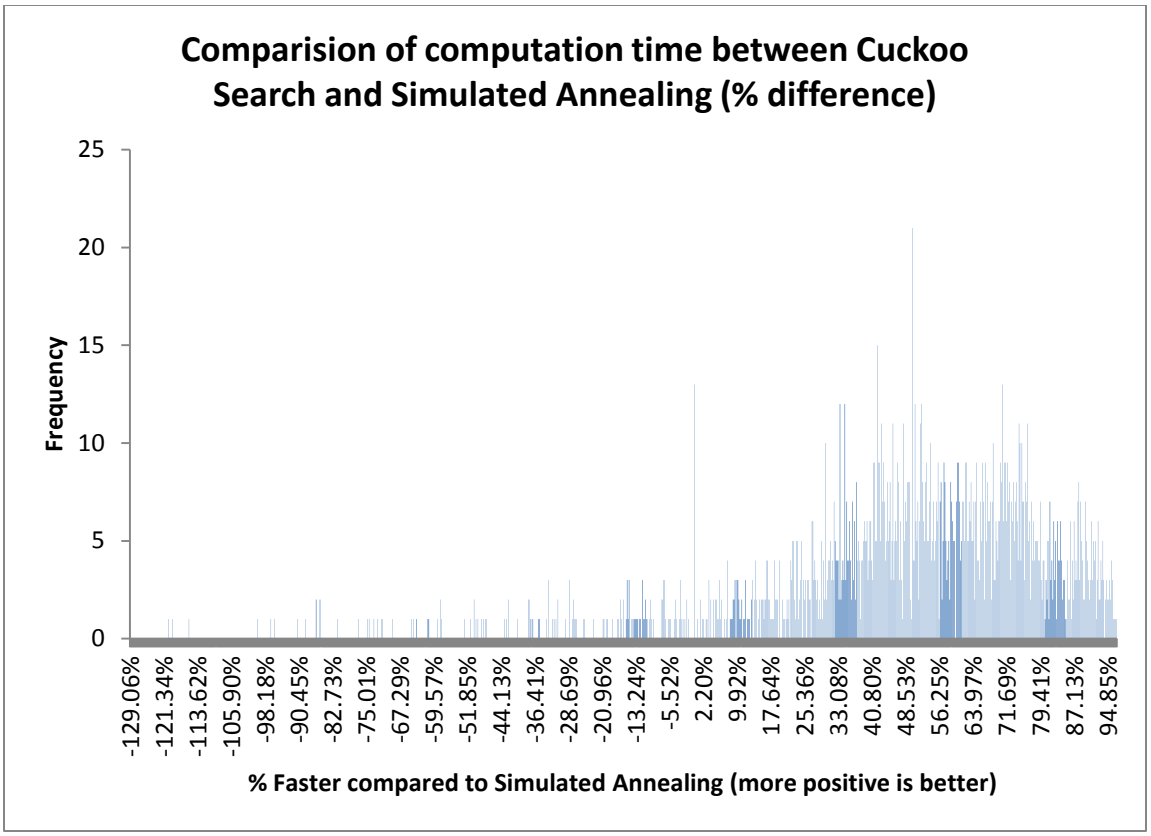

FIGURE 10 - SPEED COMPARISION, CUCKOO SEARCH VS. SIMULATED ANNEALING

Figure 10 shows each corresponding individual runs compared to Simulated Annealing. However, when computation runs are averaged first across each different files (30 run each), Cuckoo Search was very slightly slower in only 2 of them (verification5 and verification6, 0.933 and 0.927 times slower respectively) out of the total 67 files computed, and still all-in-all averaged three times faster.

7.3 ACCURACY

The accuracy of Cuckoo Search is also comparable to Simulated Annealing. Of the total 2010 different computational runs performed $\left\{\left(30\,\frac{runs}{file}\right)(67\ files)\right\}$, Cuckoo Search found a factor of safety within a 1.5% difference compared to Simulated Annealing in 1484 of the runs. Out of the 526 remaining runs, Cuckoo Search found a better-than-1.5%-difference result in 294 runs, and 232 worse results.

However, this 1.5% difference can be misleading, after all, a factor of safety of 0.02 versus 0.015 would constitute a 25% difference. Hence, the actual factor of safety difference was calculated and tabulated out of the 526 runs which did not result in a percent difference under 1.5%. A reasonable measure is a difference of 0.02 in the factor of safety found. In this case, out of the 294 runs which Cuckoo Search found a better-than-1.5%-difference result, Cuckoo Search only bettered Simulated Annealing by a factor of safety of 0.02 in 229 of them. However, of the 232 runs which Simulated Annealing bettered Cuckoo Search, the number now falls to a mere 148.

Similar comparisons were done after each 30 runs were averaged first for each individual file; the results are presented in Table 3.

Individual computation runs (Cuckoo Search vs. Simulated Annealing)			
Less-than-1.5%-difference		Less-than-1.5%-difference and greater-than-0.02 actual difference	
Total Runs	2010	Total Runs	2010
Same Results	1484	Same Results	1633
Better Results	294	Better Results	229
Worse Results	232	Worse Results	148
Success Rate*	88.46%	Success Rate*	92.64%
$\frac{better}{worse}$ ratio	1.27	$\frac{better}{worse}$ ratio	1.55
File averages (Cuckoo Search vs. Simulated Annealing)			
Less-than-1.5%-difference		Less-than-1.5%-difference and greater-than-0.02 actual difference	
Total Files	67	Total Files	67
Same Results	50	Same Results	55
Better Results	11	Better Results	9
Worse Results	6	Worse Results	3
Success Rate*	91.04%	Success Rate*	95.52
$\frac{better}{worse}$ ratio	1.83	$\frac{better}{worse}$ ratio	3
$Success\ Rate = \frac{Same + Better}{Total}$			

TABLE 3 - SUMMARY OF RESULTS, CUCKOO SEARCH VS. SIMULATED ANNEALING

Although it can be concluded that Cuckoo Search coupled with a local optimizer performed better than Simulated Annealing coupled with the same optimizer, it is important to note that aside from the test cases mentioned in Section 7.1, that in general, the surfaces found by both algorithms are extremely similar. Variations in the factor of safety can be affected by small movements of the vertices. The importance of this difference in the calculated factor of safety given two very similar surfaces should be considered alongside the massive amounts of approximations made when building the model.

The low function-evaluation count is advantageous to Cuckoo Search in that it directly translates to a higher algorithmic efficiency. However, there are a few types of situation where more function-evaluations is perhaps the only solution—this happens for example, when the landscape of the search space is littered with many delta-like functions, with overall little or no information obtainable between each individual trough.

One of such an example are slope models with a vertical drop-off face. It was found that—perhaps due to the inherent shortcomings of the method of slices and limit equilibrium—that small, geometrically and kinematically acceptable variations in the positions of vertices of a failure surface with a valid calculated factor of safety, can result in a non-valid FOS calculation using the method of slices. This worsens as the angle of the failure surface increases.

This can be seen in Figure 11. Cuckoo Search was able to find the general location of the failure surface, but with the low number of function evaluations, neither Cuckoo Search nor the post-optimization done by LCM was able to reduce the factor of safety value; in such cases, if an accurate numerical factor of safety is necessary, it is recommended that Simulated Annealing to be used.

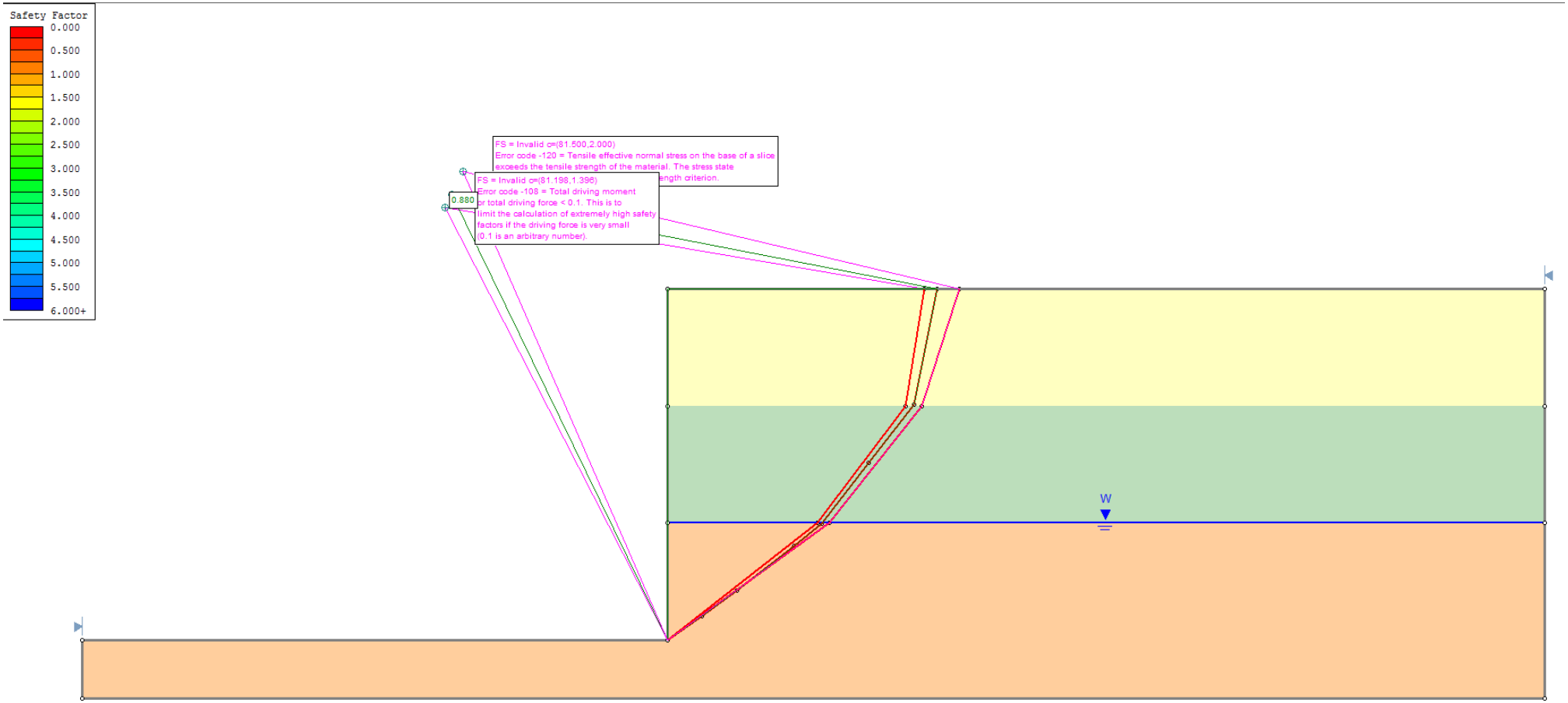

FIGURE 11 - USER15, PROBLEMS WITH LIMIT EQUILIBRIUM METHODS

7.4 THE EFFECT OF DIMENSIONALITY

The dimensionality of the problem was increased from 8 vertices (14 dimensions) to 15 vertices (28 dimensions), and 10 computational runs of each file was performed for both Cuckoo Search as well as Simulated Annealing.

The number of solutions in the solution bank for Cuckoo Search was increased from 50 to 100, while 500 iterations were kept constant. An increase of 50% computational time was observed (1.5x longer).

It was found that for cases where there is a single mode of failure (or where the global minimum was relatively easy to find), the higher number of vertices gave a finer and more detailed failure surface, allowing for a slightly lower critical factor of safety to be found. However, it suffered more in cases with local minima, such as the files described in Section 7.1.

User files 6, 9, 13, 14, with multiple failure modes (13 vertices)

User6	Local Minimum (FOS = 1.32)	Global Minimum (FOS = 1.23)
Cuckoo Search	0	10
Simulated Annealing	7	3
User9	Local Minimum (FOS = 1.23)	Global Minimum (FOS = 0.88)
Cuckoo Search	4	6
Simulated Annealing	6	4
User13	Local Minimum (FOS = 1.12)	Global Minimum (FOS = 1.82)
Cuckoo Search	5	5
Simulated Annealing	6	4
User14	Local Minimum (FOS = 0.93)	Global Minimum (FOS = 0.87)
Cuckoo Search	1	9
Simulated Annealing	4	6

TABLE 4 - HIGH DIMENSION MULTIPLE FAILURE MODE COMPARISON

Such a decrease in performance can be expected however, as doubling the dimensionality of the problem oftentimes increase the complexity by orders of magnitude.

Although the success rate for finding the true global minimum is heavily reduced, Cuckoo Search still performed better than Simulated Annealing. One should also note that on average, Cuckoo Search was faster than Simulated Annealing by over 5 times.

7.5 THE ROBUSTNESS OF THE ALGORITHM ON THIN LAYERS

Thin soil layers in a slope geometry can dictate how the slope will fail—especially if the thin layers are weak. Thin layers also present a challenge for any stochastic global optimization algorithms, as these layers are unlikely to be found given random chance. A critical failure surface passing through such a weak layer can represent a delta function in the search space, as generally speaking, little information about the weak layer is available unless the weak layer is actually found by the algorithm.

Two such weak layer files were available for testing the robustness and effectiveness of the algorithm—verification9, and user17. User17 was modified such that the thickness of the thin, weak layer was shortened to approximately 0.5", a true needle-in-the-haystack when compared to the size of the model.

Cuckoo Search was able to locate such a critical failure surface which passes through the weak layer successfully 100% of the time in both files. Typical computation results are shown in Figure 12 and Figure 13.

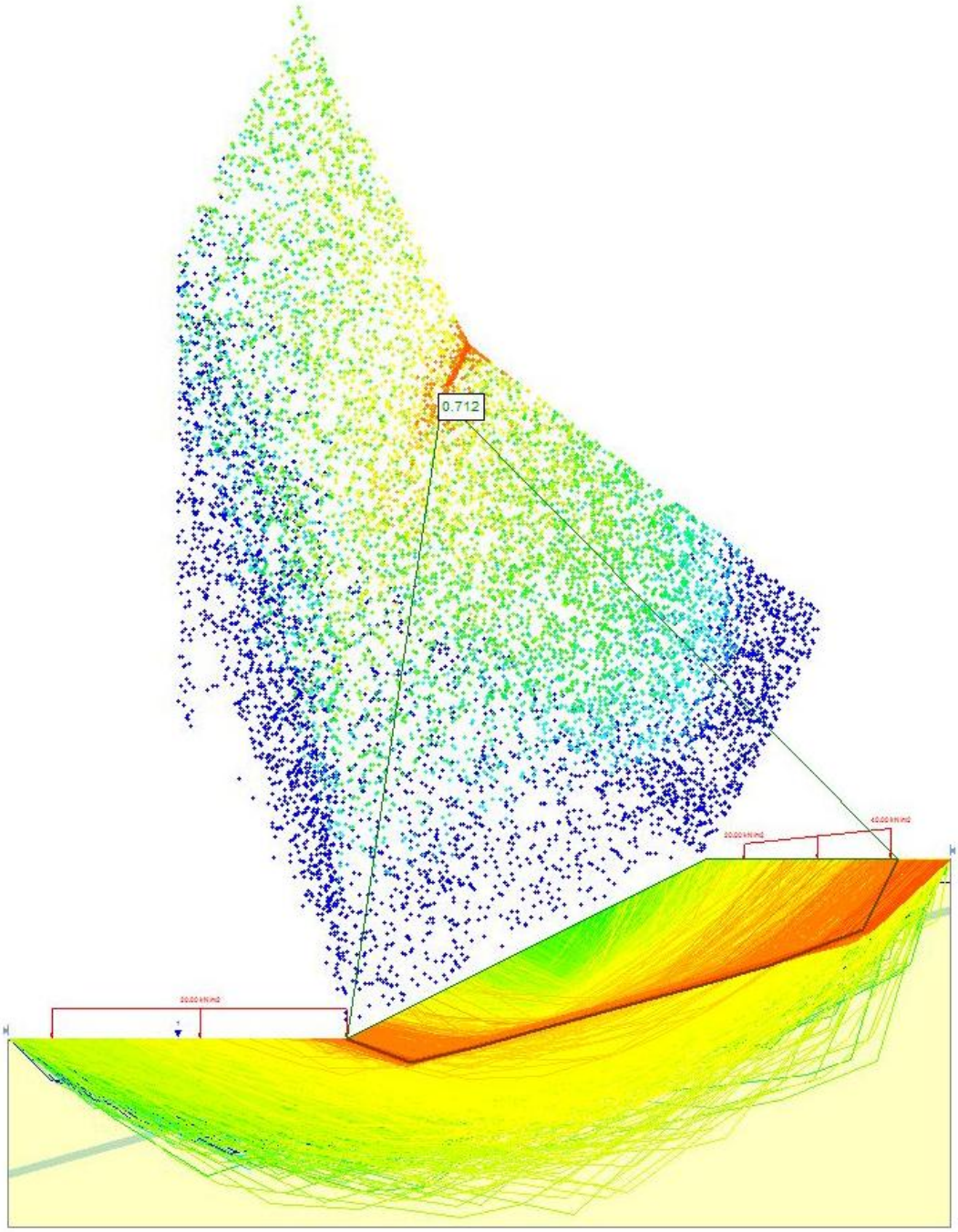

FIGURE 12 - THIN LAYER EXAMPLE, VERIFICATION9

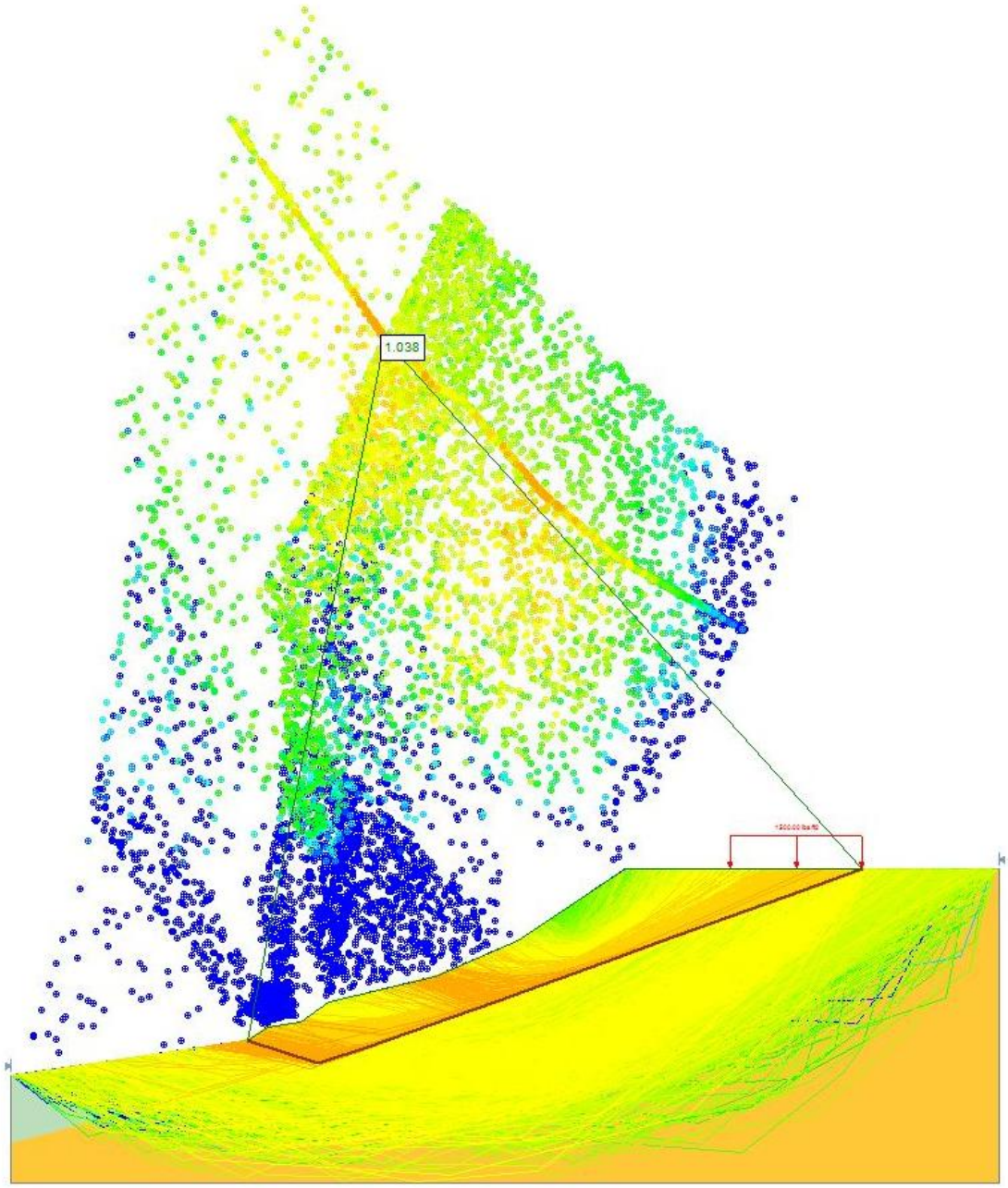

FIGURE 13 - EXTREME THIN LAYER EXAMPLE, USER17 MODIFIED

7.6 THE ROLE OF HYBRIDIZATION

As described above, Cuckoo Search was hybridized with Local Carlo Monte, a local searching algorithm. The necessity of which is described below.

The LCM is a local searching algorithm, and is extremely efficient at what it was designed to do. Thus, given a good initial guess, by for example, an experienced engineer, LCM can modify the initial failure surface to minimize its associated factor of safety quickly. Such an initial guess can be rather trivial, such as in verification1 (Figure 14), when the slope geometry is simple and soil property is non-layered and homogenous. As the complexity of slope geometry increases, guesses are often harder to make, as is the case with all of the slope geometries with multiple failure modes as described in Section 7.1. As LCM is by nature a local searching algorithm, it does not have the capability to escape local minima, and as such one will not expect LCM to find the correct failure surface given a non-optimal initial guess. It is worth noting that although the files described in Section 7.1 only contains a few prominent failure modes, that many more local minima exists—minima which LCM can easily be trapped by; the reason why those do not show up in the figures is due to the fact that Cuckoo Search can easily escape such minima, and thus do not dwell too long examining them.

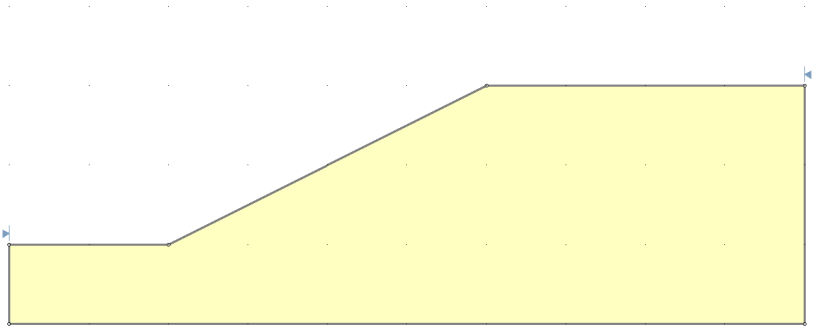

FIGURE 14 - VERIFICATION1, SIMPLE HOMOGENOUS SLOPE

The role of Cuckoo Search then becomes to provide the best "initial guess" failure surface for LCM to optimize. Of course this initial guess Cuckoo Search found will be extremely similar in shape and failure/entry/exit location to the refined surface LCM will return. It can be said that with Cuckoo Search alone, the failure surface found is oftentimes more than acceptably similar to the surface after optimization by the LCM. However, the factor of safety—the numerical measurement to the quality of the solution—at times leave more to be desired. It is important to note however, that the Cuckoo Search algorithm implemented in SLIDE is optimized more towards the efficiency and effectiveness of global searching, narrowing down the region of failure given a complex slope geometry, than to pin-point the minute difference in the numerical factor of safety value when a vertex is slightly shifted from one position to another.

Is hybridization necessary? Yes and no. LCM optimize the surface to find the lowest factor of safety. However, the accuracy of this "lowest factor of safety" depends on some other, more important aspects, such as how accurate the model representation of the actual physical slope is. Oftentimes a lot of assumptions are made when constructing a digital representation from in-situ surveys, such as homogeneity of the soil layers. Hybridization should be used if the desired failure surface to be found is one with the lowest factor of safety given the digital representation of the physical slope; depending on the accuracy of the digital model however, this may or may not be the failure surface with the actual lowest factor of safety in the physical site.

8 CONCLUSION

Cuckoo Search—a stochastic, nature-based global search algorithm—was successfully implemented in SLIDE in locating the critical failure surface of a slope geometry. Cuckoo Search was able to locate the critical failure surface more effectively (higher success rate as well as accuracy) and efficiently (lower computation time) than Simulated Annealing originally implemented to solve the exact same problem. When combined with LCM for further optimization of the critical failure surface, a refined surface with the lowest factor of safety given a slope geometry can be found.

Modifications specific to the critical failure surface searching problem are essential in increasing the performance of the search. These includes dynamic domain bounding during initialization and variation of the solutions, a suitable step-size for each control variable, as well as readjustment of surfaces after the variation of the entry and exit points but prior to the adjustments of the inner vertices.

The Cuckoo Search formulation described here was shown to have a speed increase of over three times faster than Simulated Annealing for an 8-vertices failure surface, and having a noticeable improvement in accuracy of solutions found. It triumph Simulated Annealing in locating the absolute global minimum in slope geometries with multiple failure modes; even point-wise failures. Cuckoo Search is also able to display the various failure modes found during the search, and can inform users of other possible failure modes—which the user can then explore further.

Hybridization of Cuckoo Search with LCM was necessary in the refinement of the critical failure surface for the lowest factor of safety value; Cuckoo Search itself however, is in most cases enough in locating the shape and location of the critical failure surface in the model.

Dimensionality (number of vertices in the failure surface) increase directly lead to a more complicated problem. A linear increase in the number of solutions in the solution bank with the increase in dimensions is recommended, while the number of generations is kept constant. One can see a decrease in effectiveness of locating the global minimum, and an increase in computation time, as well as precision (given that the failure surface found is truly the global minimum). Of course, with a further increase in the number of solutions in the solution bank, robustness of the algorithm can further be ensured, at the expense of computation time. Cuckoo Search under the increase in dimensionality was shown to be more effective and efficient (and more so than with lower dimensionalities) than Simulated Annealing.

9 APPENDIX I - SUMMARY FOR AVERAGES OF EACH MODEL

9.1 CUSTOMER FILES

	Cuckoo Search		Simulated Annealing		Difference	% diff btw avgs
user1.sli	average	0.75548	average	0.756019	-0.001	-0.07
	min	0.749694	min	0.749932	0.000	
	max	0.757714	max	0.757578	0.000	
	diff	0.00802	diff	0.007646	0.000	
	Std. dev.	0.00232	Std. dev.	0.001537	0.001	
	avg time	638.3667	avg time	841.5333	1.318	
	----					
user3.sli	average	1.229629	average	1.227354	0.002	0.19
	min	1.22438	min	1.22441	0.000	
	max	1.27817	max	1.2759	0.002	
	diff	0.05379	diff	0.05149	0.002	
	Std. dev.	0.00999	Std. dev.	0.009326	0.001	
	avg time	122.9333	avg time	274.7667	2.235	
	----					
user4.sli	average	0.200849	average	0.968373	-0.768	-79.26
	min	0.167441	min	0.166301	0.001	
	max	0.299249	max	1.12013	-0.821	
	diff	0.131808	diff	0.953829	-0.822	
	Std. dev.	0.053972	Std. dev.	0.155937	-0.102	
	avg time	105.6667	avg time	352.1333	3.332	
	----					
user4 revised.sli	average	0.18527	average	0.858347	-0.673	-78.42
	min	0.167446	min	0.166234	0.001	
	max	0.288278	max	0.999888	-0.712	
	diff	0.120832	diff	0.833654	-0.713	
	Std. dev.	0.040981	Std. dev.	0.293828	-0.253	
	avg time	102.2	avg time	371.2333	3.632	
	----					
user6.sli	average	1.234694	average	1.294115	-0.059	-4.59
	min	1.23177	min	1.23337	-0.002	
	max	1.23663	max	1.32101	-0.084	
	diff	0.00486	diff	0.08764	-0.083	
	Std. dev.	0.000827	Std. dev.	0.036951	-0.036	
	avg time	239.5333	avg time	607.6333	2.537	

	Cuckoo Search		Simulated Annealing		Difference	% diff btw avgs
user7.sli	average	1.135906	average	1.143577	-0.008	-0.67
	min	1.13405	min	1.13355	0.000	
	max	1.13922	max	1.37987	-0.241	
	diff	0.00517	diff	0.24632	-0.241	
	Std. dev.	0.001566	Std. dev.	0.044644	-0.043	
	avg time	194.8667	avg time	365.2	1.874	
	----					
user8.sli	average	1.724023	average	1.726036	-0.002	-0.12
	min	1.61299	min	1.71792	-0.105	
	max	1.73662	max	1.78922	-0.053	
	diff	0.12363	diff	0.0713	0.052	
	Std. dev.	0.021396	Std. dev.	0.013986	0.007	
	avg time	59.73333	avg time	190.8	3.194	
	----					
user9.sli	average	0.943403	average	1.20712	-0.264	-21.85
	min	0.871749	min	0.876407	-0.005	
	max	1.23612	max	1.21985	0.016	
	diff	0.364371	diff	0.343443	0.021	
	Std. dev.	0.130693	Std. dev.	0.062465	0.068	
	avg time	172.3	avg time	432.6	2.511	
	----					
user10 - 15m embankment dry.sli	average	1.406312	average	1.40631	0.000	0.00
	min	1.40631	min	1.40631	0.000	
	max	1.40632	max	1.40631	0.000	
	diff	1E-05	diff	0	0.000	
	Std. dev.	4.07E-06	Std. dev.	0	0.000	
	avg time	26.43333	avg time	62.43333	2.362	
	----					
user11 - dallas 5555 wall bent es9a.sli	average	1.987891	average	1.99831	-0.010	-0.52
	min	1.93589	min	1.93891	-0.003	
	max	2.03775	max	2.11349	-0.076	
	diff	0.10186	diff	0.17458	-0.073	
	Std. dev.	0.033839	Std. dev.	0.035079	-0.001	
	avg time	65.76667	avg time	118.6333	1.804	
	----					
user12 - 2550_0225.sli	average	1.077891	average	1.075794	0.002	0.19
	min	1.0674	min	1.06822	-0.001	
	max	1.09382	max	1.09297	0.001	
	diff	0.02642	diff	0.02475	0.002	
	Std. dev.	0.007805	Std. dev.	0.006357	0.001	
	avg time	40.36667	avg time	80.93333	2.005	

	Cuckoo Search		Simulated Annealing		Difference	% diff btw avgs
user13 - GH-BUCK with berm.sli	average	0.848094	average	0.856681	-0.009	-1.00
	min	0.800086	min	0.810089	-0.010	
	max	1.12004	max	1.1144	0.006	
	diff	0.319954	diff	0.304311	0.016	
	Std. dev.	0.074997	Std. dev.	0.087727	-0.013	
	avg time	102.2667	avg time	220.9333	2.160	
	----					
user14 - LOWER FAILED Buttress.sli	average	0.879649	average	0.897135	-0.017	-1.95
	min	0.866065	min	0.868994	-0.003	
	max	0.942314	max	1.34156	-0.399	
	diff	0.076249	diff	0.472566	-0.396	
	Std. dev.	0.020568	Std. dev.	0.086788	-0.066	
	avg time	107.1333	avg time	168.0667	1.569	
	----					
user15 - block search.sli	average	0.996065	average	0.903817	0.092	10.21
	min	0.939809	min	0.876841	0.063	
	max	1.05565	max	1.49941	-0.444	
	diff	0.115841	diff	0.622569	-0.507	
	Std. dev.	0.029706	Std. dev.	0.112579	-0.083	
	avg time	62.26667	avg time	669.9667	10.760	
	----					
user17 - Leachate Tank.sli	average	1.048932	average	1.115695	-0.067	-5.98
	min	1.03345	min	1.03331	0.000	
	max	1.11273	max	2.7607	-1.648	
	diff	0.07928	diff	1.72739	-1.648	
	Std. dev.	0.020505	Std. dev.	0.312305	-0.292	
	avg time	34.2	avg time	69.66667	2.037	
	----					
user20 - tstab.sli	average	3.916124	average	3.941215	-0.025	-0.64
	min	3.85142	min	3.85627	-0.005	
	max	4.05278	max	4.03428	0.019	
	diff	0.20136	diff	0.17801	0.023	
	Std. dev.	0.065412	Std. dev.	0.077736	-0.012	
	avg time	50.73333	avg time	154.7667	3.051	
	----					
user21 - slopew.sli	average	1.461281	average	1.460662	0.001	0.04
	min	1.45383	min	1.45267	0.001	
	max	1.46337	max	1.46309	0.000	
	diff	0.00954	diff	0.01042	-0.001	
	Std. dev.	0.0027	Std. dev.	0.002739	0.000	
	avg time	40.73333	avg time	68.66667	1.686	

	Cuckoo Search		Simulated Annealing		Difference	% diff btw avgs
user28 - Verifica 1 pali Rocscience moved pile.sli	average	0.766862	average	0.764834	0.002	0.27
	min	0.735337	min	0.725585	0.010	
	max	0.804414	max	0.77932	0.025	
	diff	0.069077	diff	0.053735	0.015	
	Std. dev.	0.01391	Std. dev.	0.009889	0.004	
	avg time	48.86667	avg time	463.6	9.487	
	----					
user32a - DSBarna-HF2CBMD6SC01.sli	average	0.618521	average	0.615087	0.003	0.56
	min	0.605509	min	0.607116	-0.002	
	max	0.812035	max	0.647136	0.165	
	diff	0.206526	diff	0.04002	0.167	
	Std. dev.	0.039491	Std. dev.	0.010187	0.029	
	avg time	31.43333	avg time	93.26667	2.967	

9.2 VERIFICATION FILES

	Cuckoo Search		Simulated Annealing		Difference	% diff btw avgs
verification#01.sli	average	0.983004	average	0.982963	0.000	0.00
	min	0.980784	min	0.980946	0.000	
	max	0.983885	max	0.98386	0.000	
	diff	0.003101	diff	0.002914	0.000	
	Std. dev.	0.001046	Std. dev.	0.001008	0.000	
	avg time	37.56667	avg time	61.5	1.637	
	----					
verification#02.sli	average	1.578202	average	1.577368	0.001	0.05
	min	1.57051	min	1.56923	0.001	
	max	1.58439	max	1.57979	0.005	
	diff	0.01388	diff	0.01056	0.003	
	Std. dev.	0.002612	Std. dev.	0.003106	0.000	
	avg time	30.9	avg time	61.9	2.003	
	----					
verification#03.sli	average	1.362085	average	1.362168	0.000	-0.01
	min	1.35843	min	1.35897	-0.001	
	max	1.36528	max	1.36381	0.001	
	diff	0.00685	diff	0.00484	0.002	
	Std. dev.	0.002178	Std. dev.	0.001716	0.000	
	avg time	36.4	avg time	61.13333	1.679	
	----					
verification#04.sli	average	0.980081	average	0.981806	-0.002	-0.18
	min	0.977741	min	0.977647	0.000	
	max	0.984042	max	0.994808	-0.011	
	diff	0.006301	diff	0.017161	-0.011	
	Std. dev.	0.001016	Std. dev.	0.005105	-0.004	
	avg time	45.23333	avg time	79.33333	1.754	
	----					
verification#05.sli	average	1.94753	average	1.94753	0.000	0.00
	min	1.94753	min	1.94753	0.000	
	max	1.94753	max	1.94753	0.000	
	diff	0	diff	0	0.000	
	Std. dev.	2.26E-16	Std. dev.	2.26E-16	0.000	
	avg time	36.06667	avg time	33.66667	0.933	

	Cuckoo Search		Simulated Annealing		Difference	% diff btw avgs
verification#06.sli	average	1.94753	average	1.94753	0.000	0.00
	min	1.94753	min	1.94753	0.000	
	max	1.94753	max	1.94753	0.000	
	diff	0	diff	0	0.000	
	Std. dev.	2.26E-16	Std. dev.	2.26E-16	0.000	
	avg time	35.2	avg time	32.63333	0.927	
	----					
verification#08.sli	average	1.222076	average	1.22197	0.000	0.01
	min	1.22127	min	1.22093	0.000	
	max	1.22302	max	1.22311	0.000	
	diff	0.00175	diff	0.00218	0.000	
	Std. dev.	0.000345	Std. dev.	0.00041	0.000	
	avg time	40.76667	avg time	63.23333	1.551	
	----					
verification#09.sli	average	0.715238	average	0.709313	0.006	0.84
	min	0.708452	min	0.708073	0.000	
	max	0.743109	max	0.717615	0.025	
	diff	0.034657	diff	0.009542	0.025	
	Std. dev.	0.008883	Std. dev.	0.00186	0.007	
	avg time	41.7	avg time	75.4	1.808	
	----					
verification#10.sli	average	1.493178	average	1.492079	0.001	0.07
	min	1.48923	min	1.48557	0.004	
	max	1.49729	max	1.49508	0.002	
	diff	0.00806	diff	0.00951	-0.001	
	Std. dev.	0.002305	Std. dev.	0.002623	0.000	
	avg time	53	avg time	82.43333	1.555	
	----					
verification#11.sli	average	0.759737	average	0.79792	-0.038	-4.79
	min	0.712007	min	0.709475	0.003	
	max	0.832136	max	0.894734	-0.063	
	diff	0.120129	diff	0.185259	-0.065	
	Std. dev.	0.045594	Std. dev.	0.043572	0.002	
	avg time	51	avg time	116.1333	2.277	
	----					
verification#12.sli	average	1.050334	average	1.050532	0.000	-0.02
	min	1.0399	min	1.03899	0.001	
	max	1.09914	max	1.07892	0.020	
	diff	0.05924	diff	0.03993	0.019	
	Std. dev.	0.013709	Std. dev.	0.011178	0.003	
	avg time	82.16667	avg time	633.8333	7.714	

	Cuckoo Search		Simulated Annealing		Difference	% diff btw avgs
verification#14-noncircular.sli	average	1.392185	average	1.391322	0.001	0.06
	min	1.38659	min	1.38676	0.000	
	max	1.3942	max	1.39421	0.000	
	diff	0.00761	diff	0.00745	0.000	
	Std. dev.	0.002665	Std. dev.	0.002643	0.000	
	avg time	65.16667	avg time	102.6333	1.575	
	----					
verification#15-circular.sli	average	0.41489	average	0.414518	0.000	0.09
	min	0.412005	min	0.41215	0.000	
	max	0.419129	max	0.418829	0.000	
	diff	0.007124	diff	0.006679	0.000	
	Std. dev.	0.001906	Std. dev.	0.00152	0.000	
	avg time	66.86667	avg time	137.9333	2.063	
	----					
verification#16-noncircular.sli	average	1.102323	average	1.101444	0.001	0.08
	min	1.09759	min	1.09633	0.001	
	max	1.10532	max	1.10357	0.002	
	diff	0.00773	diff	0.00724	0.000	
	Std. dev.	0.001919	Std. dev.	0.002037	0.000	
	avg time	59.96667	avg time	81.43333	1.358	
	----					
verification#19.sli	average	1.403668	average	1.403941	0.000	-0.02
	min	1.39889	min	1.39903	0.000	
	max	1.40764	max	1.40892	-0.001	
	diff	0.00875	diff	0.00989	-0.001	
	Std. dev.	0.002101	Std. dev.	0.002319	0.000	
	avg time	53.6	avg time	95.3	1.778	
	----					
verification#20-noncircular.sli	average	1.073345	average	1.081213	-0.008	-0.73
	min	1.00539	min	1.00655	-0.001	
	max	1.09364	max	1.09127	0.002	
	diff	0.08825	diff	0.08472	0.004	
	Std. dev.	0.033515	Std. dev.	0.025198	0.008	
	avg time	51.66667	avg time	81.56667	1.579	
	----					
verification#21-1.sli	average	1.990598	average	1.99122	-0.001	-0.03
	min	1.98153	min	1.9839	-0.002	
	max	1.99403	max	1.9948	-0.001	
	diff	0.0125	diff	0.0109	0.002	
	Std. dev.	0.003049	Std. dev.	0.002409	0.001	
	avg time	57.1	avg time	91.4	1.601	

	Cuckoo Search		Simulated Annealing		Difference	% diff btw avgs
verification#22-1.sli	average	1.292499	average	1.292605	0.000	-0.01
	min	1.29135	min	1.29166	0.000	
	max	1.29465	max	1.29502	0.000	
	diff	0.0033	diff	0.00336	0.000	
	Std. dev.	0.000747	Std. dev.	0.000874	0.000	
	avg time	42.33333	avg time	64.56667	1.525	
	----					
verification#24.sli	average	1.3981	average	1.397633	0.000	0.03
	min	1.39241	min	1.39445	-0.002	
	max	1.40047	max	1.39992	0.001	
	diff	0.00806	diff	0.00547	0.003	
	Std. dev.	0.001777	Std. dev.	0.001284	0.000	
	avg time	38.1	avg time	65.66667	1.724	
	----					
verification#25.sli	average	0.943173	average	0.943175	0.000	0.00
	min	0.943126	min	0.943139	0.000	
	max	0.943267	max	0.943821	-0.001	
	diff	0.000141	diff	0.000682	-0.001	
	Std. dev.	3.53E-05	Std. dev.	0.000123	0.000	
	avg time	54.86667	avg time	154.7667	2.821	
	----					
verification#27-1.sli	average	0.114236	average	0.094876	0.019	20.41
	min	0.048896	min	0.029359	0.020	
	max	0.134534	max	0.11771	0.017	
	diff	0.085638	diff	0.088351	-0.003	
	Std. dev.	0.017165	Std. dev.	0.02351	-0.006	
	avg time	29.33333	avg time	72.23333	2.463	
	----					
verification#28-Example1_Layer2.sli	average	0.014352	average	0.041039	-0.027	-65.03
	min	0.002043	min	0.001811	0.000	
	max	0.098383	max	1.09947	-1.001	
	diff	0.09634	diff	1.097659	-1.001	
	Std. dev.	0.021364	Std. dev.	0.199911	-0.179	
	avg time	53.36667	avg time	96.8	1.814	
	----					
verification#29.sli	average	2.96E-08	average	1.74E-08	0.000	70.26
	min	1.65E-10	min	3.23E-10	0.000	
	max	1.47E-07	max	1.42E-07	0.000	
	diff	1.47E-07	diff	1.42E-07	0.000	
	Std. dev.	4.09E-08	Std. dev.	3.28E-08	0.000	
	avg time	28.16667	avg time	116.5333	4.137	

	Cuckoo Search		Simulated Annealing		Difference	% diff btw avgs
verification#30-1.sli	average	1.05033	average	1.05033	0.000	0.00
	min	1.05033	min	1.05033	0.000	
	max	1.05033	max	1.05033	0.000	
	diff	0	diff	0	0.000	
	Std. dev.	4.52E-16	Std. dev.	4.52E-16	0.000	
	avg time	17.86667	avg time	61.2	3.425	
	----					
verification#31-1.sli	average	0.861257	average	0.852394	0.009	1.04
	min	0.861256	min	0.823301	0.038	
	max	0.861258	max	0.861257	0.000	
	diff	2E-06	diff	0.037956	-0.038	
	Std. dev.	4.34E-07	Std. dev.	0.014628	-0.015	
	avg time	20.86667	avg time	79.33333	3.802	
	----					
verification#32-1.sli	average	0.79942	average	0.799416	0.000	0.00
	min	0.79939	min	0.799375	0.000	
	max	0.799455	max	0.799454	0.000	
	diff	6.5E-05	diff	7.9E-05	0.000	
	Std. dev.	1.95E-05	Std. dev.	1.8E-05	0.000	
	avg time	60.86667	avg time	237.9333	3.909	
	----					
verification#41.sli	average	1.672012	average	1.671511	0.001	0.03
	min	1.66725	min	1.66691	0.000	
	max	1.67369	max	1.67345	0.000	
	diff	0.00644	diff	0.00654	0.000	
	Std. dev.	0.001813	Std. dev.	0.00205	0.000	
	avg time	268.5667	avg time	470.0333	1.750	
	----					
verification#42-noncircular.sli	average	1.868258	average	1.867964	0.000	0.02
	min	1.86606	min	1.86692	-0.001	
	max	1.87531	max	1.86951	0.006	
	diff	0.00925	diff	0.00259	0.007	
	Std. dev.	0.001845	Std. dev.	0.000835	0.001	
	avg time	27.7	avg time	61.83333	2.232	
	----					
verification#43-circ.sli	average	1.395128	average	1.363819	0.031	2.30
	min	1.31733	min	1.29494	0.022	
	max	1.52797	max	1.4859	0.042	
	diff	0.21064	diff	0.19096	0.020	
	Std. dev.	0.079534	Std. dev.	0.061447	0.018	
	avg time	45.8	avg time	700.2	15.288	

	Cuckoo Search		Simulated Annealing		Difference	% diff btw avgs
verification#44-M-C with iteration results.sli	average	0.976967	average	0.977094	0.000	-0.01
	min	0.97572	min	0.976174	0.000	
	max	0.977245	max	0.977241	0.000	
	diff	0.001525	diff	0.001067	0.000	
	Std. dev.	0.000457	Std. dev.	0.000286	0.000	
	avg time	60.23333	avg time	79.5	1.320	
	----					
verification#45-m-c.sli	average	2.7859	average	2.785547	0.000	0.01
	min	2.77992	min	2.78106	-0.001	
	max	2.78918	max	2.78918	0.000	
	diff	0.00926	diff	0.00812	0.001	
	Std. dev.	0.002331	Std. dev.	0.001694	0.001	
	avg time	26.8	avg time	48.73333	1.818	
	----					
verification#46-stage1.sli	average	2.49948	average	2.49948	0.000	0.00
	min	2.49948	min	2.49948	0.000	
	max	2.49948	max	2.49948	0.000	
	diff	0	diff	0	0.000	
	Std. dev.	4.52E-16	Std. dev.	4.52E-16	0.000	
	avg time	16.43333	avg time	61.53333	3.744	
	----					
verification#48.sli	average	0.935334	average	0.917333	0.018	1.96
	min	0.921412	min	0.911073	0.010	
	max	0.951196	max	0.93152	0.020	
	diff	0.029784	diff	0.020447	0.009	
	Std. dev.	0.007118	Std. dev.	0.005118	0.002	
	avg time	98.76667	avg time	371.3	3.759	
	----					
verification#49.sli	average	1.432501	average	1.414188	0.018	1.29
	min	1.41696	min	1.27415	0.143	
	max	1.49392	max	1.47912	0.015	
	diff	0.07696	diff	0.20497	-0.128	
	Std. dev.	0.014202	Std. dev.	0.041095	-0.027	
	avg time	46.36667	avg time	345.8	7.458	
	----					
verification#50.sli	average	0.360779	average	0.479018	-0.118	-24.68
	min	0.360778	min	0.360778	0.000	
	max	0.360781	max	1.08367	-0.723	
	diff	3E-06	diff	0.722892	-0.723	
	Std. dev.	6.75E-07	Std. dev.	0.268941	-0.269	
	avg time	23.3	avg time	140.8667	6.046	

	Cuckoo Search		Simulated Annealing		Difference	% diff btw avgs
verification#51.sli	average	0.986587	average	0.986949	0.000	-0.04
	min	0.981294	min	0.982209	-0.001	
	max	0.989856	max	0.989955	0.000	
	diff	0.008562	diff	0.007746	0.001	
	Std. dev.	0.002748	Std. dev.	0.002649	0.000	
	avg time	198.1	avg time	320.6667	1.619	
	----					
verification#52-1-dry.sli	average	2.012802	average	2.012737	0.000	0.00
	min	2.00448	min	2.00462	0.000	
	max	2.01641	max	2.01579	0.001	
	diff	0.01193	diff	0.01117	0.001	
	Std. dev.	0.00333	Std. dev.	0.003397	0.000	
	avg time	64.46667	avg time	88.36667	1.371	
	----					
verification#53.sli	average	0.758947	average	0.758498	0.000	0.06
	min	0.755846	min	0.754807	0.001	
	max	0.764887	max	0.763736	0.001	
	diff	0.009041	diff	0.008929	0.000	
	Std. dev.	0.002	Std. dev.	0.001988	0.000	
	avg time	59.73333	avg time	238.6333	3.995	
	----					
verification#54-with pile.sli	average	1.154536	average	1.155752	-0.001	-0.11
	min	1.15312	min	1.15095	0.002	
	max	1.15629	max	1.15818	-0.002	
	diff	0.00317	diff	0.00723	-0.004	
	Std. dev.	0.000817	Std. dev.	0.001792	-0.001	
	avg time	186.2667	avg time	128.1	0.688	
	----					
verification#55-slope1.sli	average	1.299496	average	1.299953	0.000	-0.04
	min	1.2925	min	1.29395	-0.001	
	max	1.30493	max	1.3019	0.003	
	diff	0.01243	diff	0.00795	0.004	
	Std. dev.	0.003639	Std. dev.	0.002707	0.001	
	avg time	54.56667	avg time	73	1.338	
	----					
verification#56-slope2.sli	average	1.293355	average	1.29193	0.001	0.11
	min	1.28733	min	1.28673	0.001	
	max	1.29869	max	1.29443	0.004	
	diff	0.01136	diff	0.0077	0.004	
	Std. dev.	0.002914	Std. dev.	0.002411	0.001	
	avg time	32.7	avg time	60.5	1.850	

	Cuckoo Search		Simulated Annealing		Difference	% diff btw avgs
verification#57-slope3-no composite.sli	average	1.37252	average	1.3726	0.000	-0.01
	min	1.36737	min	1.36886	-0.001	
	max	1.37513	max	1.37564	-0.001	
	diff	0.00776	diff	0.00678	0.001	
	Std. dev.	0.001745	Std. dev.	0.001212	0.001	
	avg time	36	avg time	59.73333	1.659	
	----					
verification#58-slope4.sli	average	0.057415	average	0.246363	-0.189	-76.70
	min	0.026834	min	0.058	-0.031	
	max	0.060343	max	0.890451	-0.830	
	diff	0.033509	diff	0.832451	-0.799	
	Std. dev.	0.005951	Std. dev.	0.333911	-0.328	
	avg time	65.9	avg time	304.0667	4.614	
	----					
verification#59-slope5.sli	average	0.025095	average	0.027417	-0.002	-8.47
	min	0.020224	min	0.01942	0.001	
	max	0.043108	max	0.092807	-0.050	
	diff	0.022884	diff	0.073387	-0.051	
	Std. dev.	0.004263	Std. dev.	0.014868	-0.011	
	avg time	49.33333	avg time	270.3667	5.480	
	----					
verification#60-slope7.sli	average	1.058633	average	1.006084	0.053	5.22
	min	1.0157	min	1.00191	0.014	
	max	1.1957	max	1.01386	0.182	
	diff	0.18	diff	0.01195	0.168	
	Std. dev.	0.067788	Std. dev.	0.003058	0.065	
	avg time	68.53333	avg time	734.5	10.717	
	----					
verification#61-m-c.sli	average	1.3645	average	1.364079	0.000	0.03
	min	1.36094	min	1.35994	0.001	
	max	1.36586	max	1.36589	0.000	
	diff	0.00492	diff	0.00595	-0.001	
	Std. dev.	0.001567	Std. dev.	0.001722	0.000	
	avg time	72.3	avg time	118.8333	1.644	
	----					
verification#62-dry-noncirc.sli	average	1.001079	average	1.001719	-0.001	-0.06
	min	0.998879	min	0.99966	-0.001	
	max	1.00253	max	1.00253	0.000	
	diff	0.003651	diff	0.00287	0.001	
	Std. dev.	0.001246	Std. dev.	0.001107	0.000	
	avg time	62.93333	avg time	104.6818	1.663	

	Cuckoo Search		Simulated Annealing		Difference	% diff btw avgs
verification#70_-_duncan_page088_figure_6-27_case1_30ft.sli	average	1.596729	average	1.595665	0.001	0.07
	min	1.59221	min	1.59193	0.000	
	max	1.59955	max	1.59722	0.002	
	diff	0.00734	diff	0.00529	0.002	
	Std. dev.	0.002002	Std. dev.	0.001449	0.001	
	avg time	23.26667	avg time	40.33333	1.734	
