A Comparison of the Robustness of Evolutionary Computation and Random Walks.

Justin Schonfeld, Bioinformatics and Computational Biology Program, Iowa State University, Ames, Iowa 50011 schonfju@iastate.edu.

Daniel A. Ashlock, Mathematics Department, Iowa State University, Ames, Iowa, 50011 danwell@iastate.edu.

Abstract—Evolution and robustness are thought to be intimately connected. Are solutions to optimization problems produced by evolutionary algorithms more robust to mutation than those produced by other classes of search algorithms? We explore this question in a model system based on bivariate real functions. Bivariate real functions serve as a well understood model system that is easy to visualize. Both the number and robustness of optimal solutions found in multiple trials with several typical optimization algorithms were compared. In the majority of the function landscapes explored the tournament selection evolutionary algorithm found optimal solutions which were significantly more robust to mutation than those discovered by the other algorithms.

I. INTRODUCTION

The Problem

Recent computer simulations of simplified protein folding using a 2-dimensional lattice model have shown that in a very specific environment “optimal” proteins found by an evolutionary algorithm are more robust to mutation than those found by a random walk [1]. This is a specific instance of a phenomenon of general interest: the way the robustness of structures depend on the methods used to create them.

This apparent ability of an evolutionary algorithm to select a more robust solution when robustness is not a fitness criteria has the potential not only to elucidate the behavior of evolution but also to aid engineers and computer scientists who tackle problems where robustness may be a critical factor. The term “robustness” is used to describe the probability a point mutation will fail to reduce the fitness of a solution. Robustness, at least the limited version investigated here, is formally defined in Section 2. In this paper we attempt to address the following two questions:

Question 1: Do evolutionary algorithms find optimal solutions which are more robust than those found by other classes of search algorithms, assuming that the robustness of the solution is not part of the fitness evaluation?

Question 2: Under what conditions do evolutionary algorithms find solutions which are more robust to mutation?

To answer these questions we compared both the quality (measured as robustness) and quantity of optimal solutions found by evolutionary and stochastic optimizers on a collection of simple optimization problems. Each algorithm was allowed the same number of fitness evaluations while optimizing several different fitness landscapes. The results were then analyzed in the context of the questions concerning robustness.

Background

Biological systems are clearly distinguished from engineering systems in a number of ways. Biological systems are less predictable, more able to tolerate variation in their inputs and operating environment, and often are able to fail more gracefully. A half squashed ant, for example, can continue to walk while a half squashed car seldom functions at all. Several of the qualities that distinguish biological and engineering systems fit under the rubric of “robustness”. It is tempting to ascribe the robustness of biological systems to their origins via evolution, but first the term robustness must be defined.

There are an enormous number of different types of robustness. Robustness might be defined as retaining function in the face of mutation, e.g. for a protein sequence. The ability of a person to continue navigating with mud in one eye is another type of robustness. The
ability to survive in a broad variety of climates is yet another notion of robustness. The type of robustness we wish to study is closest to the first example. We will study the ability of data structures produced by a stochastic search system, using some collection of search operators, to retain their quality in the face of additional application of those search operators.

How might using evolution, instead of other stochastic search techniques, produce robustness? Evolution operates on populations of structures. As a population loses diversity its members become similar. Evolutionary computation systems, which usually feature panmictic breeding and small populations, lose diversity rapidly. Once the system has settled into a state where most of the population have the same locally high fitness, secondary selection to resist the stochastic search operators begins to take place. Imagine that we have a population with many members on a plateau of the fitness landscape. Mutation moves creatures either within or off of the edges of the plateau. Those farthest from the edge are more likely to have children that remain on the plateau. This, in turn, suggests that the population will pile up away from the edges of the plateau. The robustness of a population member is his distance from the edge of the plateau. If a population member is not on a plateau then he has no robustness. Only this limited form of robustness just outlined is addressed in this study.

The various forms of evolutionary computation such as evolutionary programming, genetic algorithms, and evolution strategies all use some version of the biological paradigm of evolution and hence resample near good solutions. This means that the notion of basins of attraction about an optima, while less crisp than in the case of gradient following optimizers influences which solutions are more likely to be found. This paper seeks to document and quantify effects related to the size, shape, and number of basins of attraction.

For the observation that inspired this research, the exceptional stability of biological proteins and their lattice analogs located by evolutionary algorithms, the notion of robustness against mutation operators (in biology and as the search operators in the lattice analogs) is close to the notion of robustness used in this study. The large plateaus created as topologically interesting optima in our bivariate function model system are analogous to functionally equivalent variants of a protein in which mutations are made to non-critical protein residues.

The novel contribution of this work lies in the investigation of the effect of the shape and juxtaposition of distinct optima and their basins of attraction. In applied evolutionary search problems the effect of shape is probably significant, but documenting shape is difficult. The use of self adaption in many evolutionary search strategies [2] shows that the spaces are not isotropic. The fact that adapting the shape of search, even to first order, helps document a need to understand the effect of shape. This study takes one possible step into this arena.

II. Model Specifications

To answer the questions posed in the introduction we compared the robustness of solutions produced by two different evolutionary algorithms to those produced by a random walk and a stochastic hill climber on a variety of fitness landscapes. We used fitness landscapes described by bivariate functions, because the optimization of such landscapes by evolutionary algorithms has been previously studied and is relatively well understood [3], [4]. The use of two dimensional surfaces made visualization of the resulting population distributions possible.

Bivariate Function Model System

The fitness of a solution, a pair of (X,Y) coordinates, was determined by evaluating the landscapes function at the solutions (X,Y) coordinates. Each landscape was trimmed by a pair of parallel planes oriented perpendicular to the z-axis, called the upper and lower trimming planes (UTP and LTP respectively). Any solution which evaluated to a height greater than that of the upper trimming plane, returned the height of the upper plane as its fitness. A solution with a height smaller than that of the lower trimming plane was assigned a fitness of 0.

This trimming procedure changed each landscape into a series of plateaus with surrounding basins of attraction. Each plateau represents all of the possible instances of a single optimal solution within the fitness landscape. By raising and lowering the upper trimming plane we controlled the size of each plateau and its surrounding basin of attraction.

Evolutionary Algorithms

The evolutionary algorithms we used were a single tournament selection algorithm and a variation of the great deluge algorithm [5], [6]. The single tournament
Great Deluge Algorithm

01. Initialize population of solutions
02. Initialize a lower bound $b$ at 0 fitness
03. Do $g$ times
04. Remove all solutions with fitness less than $b$
05. Create children by selecting from the remaining population uniformly at random
06. Mutate each child
07. Record fitness for each child
08. Increase $b$

Stochastic Hill Climber Algorithm

01. Initialize vector of solutions
02. Do $g$ times
03. Mutate each solution in the vector
04. If the new solution has a fitness $\geq$ the old solution save it
05. Record fitness for each solution

Tournament Selection Algorithm

01. Initialize population of solutions
02. Do $g$ times
03. Shuffle the population
04. Divide the population into families of size $t$
05. Copy the $t/2$ most fit from each family into children
06. Mutate each child
07. Record fitness for each child

Great Deluge Algorithm

01. Initialize population of solutions
02. Initialize a lower bound $b$ at 0 fitness
03. Do $g$ times
04. Remove all solutions with fitness less than $b$
05. Create children by selecting from the remaining population uniformly at random
06. Mutate each child
07. Record fitness for each child
08. Increase $b$

Tournament Selection Algorithm

01. Initialize population of solutions
02. Do $g$ times
03. Shuffle the population
04. Divide the population into families of size $t$
05. Copy the $t/2$ most fit from each family into children
06. Mutate each child
07. Record fitness for each child

Great Deluge Algorithm had a tournament size of $t=4$. The great deluge algorithm raised the lower bound, $b$, by a constant amount each generation beginning at 0 and ending at slightly below the optimal fitness. Mutation was implemented by picking a number from a Gaussian distribution with a mean of 0 and a std. dev. of 1. The random number selected was then multiplied by a mutational coefficient ($\mu$) ranging from 0.25 to 1.0 to scale it, before being applied to the child (both coordinates were mutated independently).

Non-Evolutionary Stochastic Algorithms

Random Walk Algorithm

01. Initialize a vector of solutions
02. Do $g$ times
03. Mutate each solution in the vector
04. Record fitness for each solution

The random walk (RW) algorithm was used primarily as a control. Each random walk was simulated by choosing a random starting point (coordinate pair) in the fitness landscape, and then mutating that initial point $g$ times. The hill climber algorithm (HC) also started at a random point, but moved only if the new solution was at least as fit as the current one. Each algorithm was performed $p$ times.

Analysis

To determine which algorithm produced more robust optimal solutions both the terms optimal solution, and robustness required strict definitions.

Definition II.1. Optimal Solution Each coordinate pair with a fitness equal to that of the upper trimming plane is an optimal solution. This amounts to having coordinates within one of the regions affected by the trimming plane. Two coordinate pairs on the same plateau within a landscape represent separate instances of the same optimal solution. Coordinate pairs on distinct plateaus represent distinct optimal solutions.

Definition II.2. Robustness The robustness of a solution is the probability that a single mutation will fail to transform an optimal solution into a non-optimal solution or a different optimal solution. Thus $0 \leq \text{Robustness} \leq 1$.

To determine the shape and connectedness of each plateau, the XY-plane of each fitness landscape was divided into a grid. The grid was then subjected to a recursive fill algorithm that labeled squares containing optimal solutions with the unique identifier of the plateau they belonged to. Each grid square was represented by the point at its center. This procedure enabled us to identify and label distinct plateaus. The robustness for each grid square was calculated by performing $r$ random mutations on the mid-point of the square and counting the number of results that fell within the same optimal solution (plateau).

In addition to computing the average number of times each optimal solution was found, we also calculated the average robustness of the optimal solutions located by each algorithm. The average robustness ($Q$) was calculated for each distinct optimal solution (plateau $p$) using the following equation:

$$Q(p) = \frac{\sum_{(x,y) \in P} \text{rob}(x,y) \ast \text{no.hits}(x,y)}{\text{no.hits}_{p}}$$  \hspace{1cm} (1)$$

The term, no. hits, used in relation to an instance $(x,y)$ of an optimal solution describes the number of times that instance was found. When used in relation to a distinct optimal solution $p$ the term refers to the number of times any instance of that optimal solution
was found. The average robustness of the instances of an optimal solution \( Q \) found by an algorithm are referred to as the quality of the robustness to distinguish it from the constant valued average robustness of each plateau.

### III. Experiments

#### Landscapes

<table>
<thead>
<tr>
<th>ID</th>
<th>( \mu )</th>
<th>Area</th>
<th>Perim.</th>
<th>Tot. Ro.</th>
<th>Avg. Ro.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SH</td>
<td>1.00</td>
<td>1251</td>
<td>112</td>
<td>407.230</td>
<td>0.326</td>
</tr>
<tr>
<td>SH</td>
<td>0.75</td>
<td>1251</td>
<td>112</td>
<td>562.680</td>
<td>0.450</td>
</tr>
<tr>
<td>SH</td>
<td>0.50</td>
<td>1251</td>
<td>112</td>
<td>767.410</td>
<td>0.613</td>
</tr>
<tr>
<td>SH</td>
<td>0.25</td>
<td>1251</td>
<td>112</td>
<td>1002.770</td>
<td>0.802</td>
</tr>
<tr>
<td>TSH</td>
<td>1.00</td>
<td>626</td>
<td>78</td>
<td>124.500</td>
<td>0.199</td>
</tr>
<tr>
<td>TSH</td>
<td>1.00</td>
<td>626</td>
<td>78</td>
<td>123.520</td>
<td>0.199</td>
</tr>
<tr>
<td>CRA</td>
<td>1.00</td>
<td>1252</td>
<td>220</td>
<td>291.490</td>
<td>0.233</td>
</tr>
<tr>
<td>CRE</td>
<td>1.00</td>
<td>1253</td>
<td>194</td>
<td>304.200</td>
<td>0.243</td>
</tr>
<tr>
<td>HCRE</td>
<td>1.00</td>
<td>1257</td>
<td>194</td>
<td>306.169</td>
<td>0.244</td>
</tr>
<tr>
<td>HCRA</td>
<td>1.00</td>
<td>1258</td>
<td>220</td>
<td>297.920</td>
<td>0.237</td>
</tr>
<tr>
<td>HCRA</td>
<td>1.00</td>
<td>1250</td>
<td>112</td>
<td>408.630</td>
<td>0.327</td>
</tr>
</tbody>
</table>

*Table III.1.* Descriptive statistics for each landscape.

The experiments described in this paper each use one or more of the following fitness landscapes: single hill, two small hills, crater, crescent, hill and crescent, and hill and crater. The single hill landscape was used to calibrate the model and test the behavior of key model parameters. The double hill landscape was used to test for interaction between plateaus. The crater and crescent landscapes were used to try to understand the effect of shape on the quality of robustness. Finally, the composite landscapes were used to explore the effects of shape on the interaction between peaks in multi-peak landscapes.

Each landscape contained at least one optimal solution (plateau). Each plateau was characterized by its area (in grid units), perimeter length (the number of grid squares belonging to the plateau which bordered on less fit solutions), total robustness (sum of the robustness of each grid square on the plateau), and average robustness (total robustness / area).

The mutational coefficient term \( (\mu) \) represents the variance of the standard normal distribution used to perform mutation. For each value of the mutational coefficient all robustness values were determined by iterated sampling. As can be seen in Table III.1, by decreasing the variance (and therefore the impact) of mutations the robustness of each instance of an optimal solution is increased. The characteristics of each landscape are summarized in Table III.1. Algebraic and visual representations for each landscape, after trimming, appear in Table III.2.

#### Default Parameters

The default parameters used for each of the experiments described in this paper are listed in Table III.3. Variation from the default parameter values is specified in the individual descriptions of each experiment.

Number of Trials specifies the number of times each experiment was repeated. The Tournament Size parameter indicates the number of individuals which compete against each other in each tournament during a fitness evaluation. The Population Size and Number of Generations parameters are used for both the evolutionary algorithm and the random walk to determine the number of potential solutions each algorithm generates.

The parameters X Origin, X Size, Y Origin, Y Size,
### Table III.3. Default model parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Trials</td>
<td>1000</td>
</tr>
<tr>
<td>Population Size ($p$)</td>
<td>20</td>
</tr>
<tr>
<td>Tournament Size ($t$)</td>
<td>4</td>
</tr>
<tr>
<td>Number of Generations ($g$)</td>
<td>50</td>
</tr>
<tr>
<td>X Origin</td>
<td>-5.0</td>
</tr>
<tr>
<td>X Size</td>
<td>10.0</td>
</tr>
<tr>
<td>Y Origin</td>
<td>-5.0</td>
</tr>
<tr>
<td>Y Size</td>
<td>10.0</td>
</tr>
<tr>
<td>Grid Size</td>
<td>0.5</td>
</tr>
</tbody>
</table>

and Fill Size bound the landscapes and describe the grid used to divide up the XY-plane of each fitness landscape. X Origin and X Size determine the domain of the X coordinate, and Y Origin and Y Size do the same for Y. Fill Size determines the length and width of each grid square.

The default upper and lower trimming planes for each landscape were set so as to create optimal solution plateaus with the same area for landscapes which are compared with one another.

**Experimental Results**

**Experiment 1: The effect of the mutational coefficient:** The first experiment explores the effect of altering ($\mu$) as the algorithms search the single hill landscape for optimal solutions. Each algorithm was run with four separate values of ($\mu$): 0.25, 0.50, 0.75, and 1.00. The results are given Figure III.1. At a low mutational coefficient (0.25 to 0.50) the tournament selection algorithm, on average, found the optimal solution more often. At higher values of ($\mu$) the hill climbing algorithm was more successful at finding the optimal solution. The great deluge algorithm had the highest variance, but was less effective than any of the other algorithms, excepting the random walk. Raising ($\mu$) increased the number of hits made by the random walk by a small degree. At low values of ($\mu$) (0.25 to 0.75) tournament selection found significantly more robust solutions than the other algorithms. As ($\mu$) increased the robustness of the solutions found by each of the four algorithms converged.

**Experiment 2: Effect of reducing support:** Experiment 2 explores the effect of raising the lower trimming plane.

As the lower trimming plane is raised the average number of hits for each algorithm decreased, with the exception of the random walk which did not change. As the lower trimming plane increased from 0 to 0.5 the hill climber, which initially found the optimal solution most often, changed places with tournament selection. The great deluge algorithm was significantly less effective than either the hill climber or the tournament selection, but more effective than the random walk (Figure III.2).

This suggests after an initial search for an optimal solution the evolutionary and hill climbing algorithms settle into an equilibrium state which does not depend on reacquiring the plateau. If true, this should have an impact in multiple-plateau landscapes.

Reducing the basin of attraction had little significant effect on the quality of robustness obtained by any of the algorithms. Tournament selection, again, found the most robust solutions, followed by the hill climber, and the random walk. The variance of the average robustness values obtained by the great deluge method, however, increased dramatically as the basin of attraction was shrunk. The large variance in behavior of the great deluge made analysis challenging.

**Experiment 3: The effect of support on determining interaction in multiple solution landscapes:** Experiment 3 examined the effect of raising the lower trimming plane in multi-plateau landscapes. The double hill landscape, which contains a pair of optimal solutions identical in size and shape, is used for this experiment.

The hill climber algorithm finds the optimal solution significantly more often than the other methods. Unlike the single hill landscape, while the tournament selection and hill climber algorithms converge, they do not change places. As in Experiment 2 the great deluge algorithm finds the optimal solutions significantly more often than the random walk, but significantly less than the other two algorithms. Splitting the single hill landscape into two smaller hills has no significant effect on the ordering of the quality of robustness of the optimal solutions located by each algorithm.

**Experiment 4: Optimal Solution Shape:** The fourth experiment explores the effect of the shape of the optimal solution space. Three different landscapes were used, each with a single optimal solution, but also each with a distinct perimeter. The single hill landscape had the smallest perimeter and the highest average robustness. The crescent landscape had the middle perimeter value and a much lower average robustness. The final landscape, the crater, had the greatest perimeter value, and the smallest average robustness value.
Figure III.1. The effect of altering ($\mu$) from 0.25 to 1.0. Left: Average number of hits found by each algorithm with 95% confidence interval. Right: Average quality of robustness found by each algorithm with 95% confidence interval.

Figure III.2. The effect of raising the lower trimming plane on the robustness and number of optimal solutions found on the single hill landscape. Left: Average number of hits found by each algorithm with 95% confidence interval. Right: Average quality of robustness found by each algorithm with 95% confidence interval.

Figure III.3. The effect of raising the lower trimming plane in the double hill landscape. Left: Average number of hits found by each algorithm with 95% confidence interval. Right: Average quality of robustness found by each algorithm with 95% confidence interval.
Figure III.4. The effect of shape on the performance of each algorithm. The landscapes are arranged from right to left in decreasing order according to the perimeter/surface area ratio of their plateau. Left: Average number of hits found by each algorithm with 95% confidence interval. Right: Average quality of robustness found by each algorithm with 95% confidence interval.

The shape of the optimal solution space had a statistically significant effect on the number of hits and on the quality of robustness of those hits. The number of hits obtained by the random walk remained relatively independent of the shape of the plateau. The number of hits obtained by tournament selection, on the other hand, appeared to be correlated to the perimeter/surface area ratio of the plateau. The hill climber, which found the optimal solution most often in each landscape, also appeared to be significantly affected by the shape of the landscape. The hill climber found the optimal solution significantly less often on the crescent landscape than it did on either the single hill or the crater.

The quality of robustness obtained by each algorithm also varied significantly depending on the shape of the landscape. On the single hill and crescent landscapes tournament selection found instances of the optimal solution which were, on average, significantly more robust than those found by the other techniques. On the crater landscape, though, while tournament selection still significantly better, it is by a much smaller degree.

Experiment 5: Composite Landscapes: The fifth experiment examines multi-solution landscapes, of which Experiment 3 was an example, in which the shape of the solutions is not identical. The two landscapes used were the hill and crescent landscape, and the hill and crater landscape(Figure III.5).

IV. DISCUSSION AND CONCLUSIONS

The size of the mutation, the size of the basin of attraction around the optima, and the number and shape of optima all have an impact on the behavior of both the evolutionary algorithm and non-evolutionary algorithms. As expected all of the algorithms found optimal solutions more reliably than the random walk by a large (and statistically significant) degree. The hill climber competes with the tournament selection algorithm for top honors in number of solutions found as the character of mutation is varied. As the strength of the mutation is increased the hill climber finds the optimal solution more often, most likely because it is able to search a larger area. The tournament selection algorithm, on the other hand, suffers from a larger $\mu$ which keeps pushing the population off of the optimal solution plateau. In all but a few cases the evolutionary algorithm also found solutions which have a significantly higher quality of robustness than those obtained by either of the non-evolutionary algorithms. The great deluge and random walk both turned in terrible performances on the problems examined.

It is interesting to compare Experiments 3 and 5. These experiments both used landscapes with multiple plateaus. The distinction based on solution robustness between the random walk and evolutionary algorithm narrowed when the landscapes were made more complex. Recalling the no free lunch theorem [7], it must be that case that there are landscapes where the random walk is the superior algorithm. One would expect these cases to be the more nearly random spaces, perhaps slightly approximated by our more complex landscapes.

In a landscape with multiple nearby plateaus there is a chance of a stochastic optimizer losing and rediscovering optima. Returning to the original motivation
for this work, understanding the substantial stability of biologically evolved proteins, there is an issue that may confound attempts to understand choices between functionally equivalent structures. The shape of a plateau (region of mutationally connected acceptable protein function) in several hundred dimensions (one per amino acid residue) is quite difficult to apprehend. A family of proteins that can be used to mutually rediscover one another over the course of evolution could represent an source of selective advantage that would be extremely difficult to detect.

V. Future Work

Given the intriguing results on shape a logical course for future investigation is to perform studies on more shapes, pairs of shapes, and sets of shapes. Such a study could include control for an exploration of the parameters of mutational diameter and mutational separation. Creating families of “related” plateaus that are well separated from one another would be an interesting case, especially if a statistical test distinguishing the behavior of families of plateaus from single plateaus could be constructed.

The evolutionary and non-evolutionary algorithms examined here are, aside from the change of mutation variance, just a few examples of rich collections of algorithms. A survey of different EA techniques to document the degree of robustness they exhibit might be a valuable line of study. It may also be that the notion of robustness adds an additional feature to the exploration/exploitation trade-off. Additional exploitation may yield gains in robustness.

The fitness landscapes used in this study are simple and easy to visualize. It is possible to extend the work to more challenging landscapes and types of structures. In [8] the very complex case of evolved computer code is studied and it is found that high mutation rates force an evolutionary system into a broad, flat optima. Filling in the gap between this study and studies on highly complex representations like evolvable code is a necessary step in applying the results to the design of evolutionary computation systems.

REFERENCES