A Study of Mutational Robustness as the Product of Evolutionary Computation

Justin Schonfeld
University of Nevada, Reno
MS 171
Reno, Nevada
schonfju@cse.unr.edu

ABSTRACT

This paper investigates the ability of a tournament selection based genetic algorithm to find mutationally robust solutions to a simple combinatorial optimization problem. Two distinct algorithms (a stochastic hill climber and a tournament selection based GA) were used to search for optimal walks in several variants of the self avoiding walk problem. The robustness of the solutions obtained by the algorithms were compared, both with each other and with solutions obtained by a random sampling of the optimal solution space. The solutions found by the GA were, for most of the problem variants, significantly more robust than those found by either the hill climbing algorithm or random sampling. The solutions found by the hill climbing algorithm were often significantly less robust than those obtained through random sampling.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms

Keywords

Genetic Algorithms, Robustness, Self Avoiding Walk, Tournament Selection

1. INTRODUCTION

DEFINITION 1.1. **Robust** - *1. Strong and healthy; hardy; vigorous*

Robustness is a valuable property, both in biological organisms and man made systems. A robust genetic code allows an organism to reproduce in a world rich with mutagenic agents and changing environmental conditions. A robust design allows buildings to weather earthquakes and hurricanes. A robust immune system helps people survive infection and other illness.

The study of robustness in nature is a fast growing area of both molecular biology and population genetics [13, 14]. From the triplet nature of the genetic code to the hardiness of bacterial metabolic networks [5] it can be seen again and again that naturally evolved systems are robust.

Robustness, as a product of evolution, is not limited to just naturally evolved systems. 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 [10]. In [15] the very complex case of evolved computer code is studied and it is found that high mutation rates force an evolutionary system into broad, flat optima as opposed to high, narrow optima.

Genetic algorithms have been used to solve a number of challenging engineering problems. Genetic algorithms have been used successfully to design circuit diagrams [6], baffle placement in stoves [11], and stable systems controllers [7].

This study builds on work by Schonfeld and Ashlock [8, 9] to explore the mechanism for the location of robust optimal solutions by genetic algorithms. If the mechanism by which genetic algorithms develop robust solutions can be successfully isolated then it can be exploited to develop robust solutions to real world optimization problems.

In this paper only a specific type of robustness is considered, the mutational robustness of optimal solutions. Informally, the mutational robustness of an optimal solution is the probability that a single point mutation will render the solution non-optimal. That is, the probability that the mutation will reduce the fitness of the optimal solution to below the maximum fitness. The terms *optimal solution* and *robustness* are defined more formally in the context of the Self Avoiding Walk (SAW) problem in Section 2.

It should be noted that the concept of mutational robustness is not new to this work or the previous work by Schonfeld et. al. In [12] E.Nimwegen et. al. proposed a general model for a population evolving on a neutral network, that is, for a population evolving on a set of solutions with equivalent fitness which are connected by single point mutation. The success of the genetic algorithm at finding robust optimal solutions is predicted by their analysis of their model. The topic of mutational robustness has also been examined in the context of evolutionary computing by Bullock in [3]. Bullock explored the manner in which a genetic algorithms

GECCO'07 July 7–11, 2007, London, England, United Kingdom.

tendency to select mutationally robust solutions affects how it explores neutral networks. Unfortunately the author did not become aware of these preceding works until late in the final revision process of this paper and was unable to modify this paper to adequately reflect the contributions of these works. The reader is referred to both of them for an excellent treatment of similar material.

In Schonfeld et. al., 2004, [8] an evolutionary algorithm using tournament selection was compared to the great deluge algorithm[1], a stochastic hill-climber, and a random walk. For several functional optimization problems the tournament selection algorithm was found to produce optimal solutions which were significantly more robust to point mutation than those of the other algorithms, except the great deluge. The great deluge was often, but not always beaten by the tournament selection algorithm. Its performance sometimes failed to be significantly different from that of the tournament selection algorithm because of a very high variance in the robustness of the structures located by the great deluge algorithm,

A second study by Schonfeld et. al., 2005,[9] investigated a slightly different question: given an evolutionary optimization problem with many possible genotypes for each phenotype are the evolved genes for a given phenotype more robust to point mutation than randomly sampled genes for the same phenotype. They answered the question using a cellular representation for polyominos in the plane. The evolutionary computation system optimized for shapes which packed well onto the surface of a torus when dropped at random. For the majority of the evolved phenotypes the evolved genes for a given shape proved to be significantly more robust to point mutation than those sampled at random for that same shape. A few evolved genotypes, however, were not significantly more robust than those sampled at random and in some cases were less robust.

Building on the conclusions of these previous studies, that evolutionary algorithms do indeed evolve more robust solutions in several different model systems, this study investigates the character of that behavior more fully. For that purpose the Self Avoiding Walk problem, where the ability of a walk to completely cover a 2D grid is optimized, was chosen. This problem has several advantages over both of the previous optimization problems: it is more transparent than the polyomino tiling problem, it is more challenging than the function optimization problem, it can be tuned to provide increasing levels of difficulty, and the optimal solutions can be easily enumerated and their exact robustnesses calculated.

The remainder of this paper is divided into four sections: Section 2 describes the SAW problem and the experimental design, Section 3 gives the results of the experiments, Section 4 contains the conclusions, and Section 5 the future directions.

2. THE SAW PROBLEM AND EXPERIMENTAL DESIGN

2.1 The SAW Problem

2.1.1 Definition of the SAW Problem

A self avoiding walk is a walk on a grid which visits every grid square without visiting the same square more than once. The SAW problem is defined as follows:

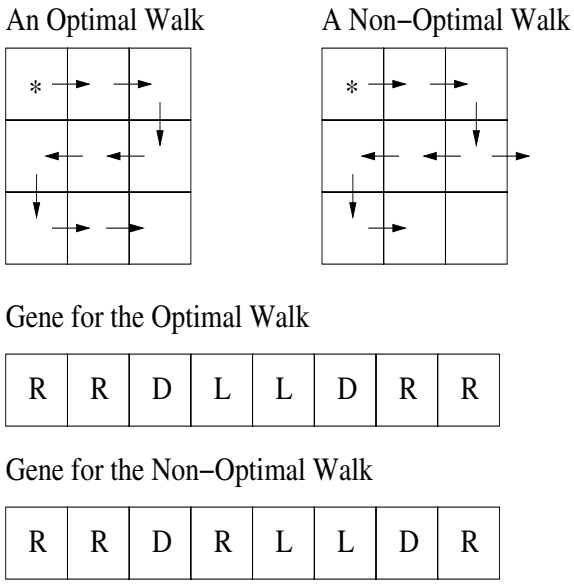

Figure 1: Examples of both optimal and non-optimal solutions to the 3x3 Self Avoiding Walk problem with a representation length of 8. In each example the walk starts in the upper left corner and grows according to the directions in the gene.

Given: A grid of size M x N, a fixed starting position, and enough instructions to permit visiting all of the grid squares.

Find: A set of instructions which generates a path that visits every grid square.

If the number of instructions allowed is equal to MN-1 then each optimal solution is a self avoiding walk. If a walk with more instructions visits every grid square then it still solves the problem optimally, but it is not necessarily self avoiding.

Each walk is represented as a string over the alphabet $\{U, R, D, L\}$. Each character corresponds to one of the four directions available on a 2D grid: up, right, down, and left. The length of the string is referred to hereafter as the *representation length*, and the string itself is referred to as the *gene*. A gene is evaluated by growing the walk from a fixed starting point, in the upper left corner of the grid, according to the directions in the gene read from left to right. Instructions which would grow the walk beyond the limits of the grid are ignored. Examples of both optimal and non-optimal walks are given in Figure 1.

The fitness of the self avoiding walk (SAW) is scored by counting the number of grid squares visited by the walk. The minimum possible fitness for a walk is one, and the maximum possible fitness is MN.

In the context of the SAW problem the terms *optimal solution* and *robustness* are defined as follows.

DEFINITION 2.1. **Optimal Solution** *A walk which visits every square in the grid.*

DEFINITION 2.2. **Robustness** *The robustness of a structure is the probability a mutation will not change it's fitness. Since the mutation operator allows the possibility of a null mutation at a rate of* $\frac{1}{4}$*, 0.25* $\leq$ *robustness* $\leq$ *1.*

2.1.2 Properties of the SAW Problem

One of the advantages of using the SAW problem as a model system is the ability to quickly (for small grid sizes) and easily enumerate each of the optimal solutions and all of their mutants. This allows one to estimate both the difficulty of the particular SAW problem being solved as well as the potential in the solution space for robustness.

Several basic properties of the SAW problem variants used in the following experiments are displayed in Table 1. Two trends are immediately apparent from reading the table: first, the fraction of the solution space occupied by optimal solutions increases with the length of the representation and second, the range of the robustness values of the optimal solutions also increases with the length of the representation.

The first trend is easily explainable by considering how the optimal solution space and search space grow as the length of the representation is increased. The size of the search space is given by l^4 where, l is the length of the representation. Therefore, each additional instruction increases the search space by exactly a factor of four. For any given optimal solution four new optimal solutions can be generated simply by adding an extra instruction to the end of the solution. An additional $2M + 2N - 2$ optimal solutions can be generated by inserting the extra instruction so it directs the walk to try to build into a wall. This means that for each increase in instruction length the number of optimal solutions grows by at least a factor of $2 + 2(M + N)$.

The second trend can be explained in light of the first. The robustness is scored as a ratio of the number of 1-mutant optimal solutions divided by the number total number of 1-mutants. Adding an instruction to an optimal solution increases the number of 1-mutant optimal solutions by $2 + 2(M + N)$ while only increasing the number of mutants by 4.

2.2 Search Algorithms

This study compares two distinct search algorithms: a stochastic hill climber, and a tournament selection based genetic algorithm. The algorithms are compared both with each other and against a baseline average determined by selecting from the pool of optimal solutions uniformly at random.

Each algorithm was implemented so that it had the same number of fitness evaluations. Both the hill climber and the genetic algorithm used the same mutation operator: (1) m instructions were chosen without replacement from the representation, and (2) each of the chosen instructions was replaced with a new direction chosen from the allowable four uniformly at random. There is, therefore, a one in four chance that the new instruction will be the same as the old instruction.

The average robustness score for a single run of an algorithm was calculated as follows:

$$\frac{\sum_{i=1}^{nopt} found(i) \cdot rob(i)}{\sum_{i=1}^{nopt} found(i)},$$

where $nopt$ is the number of optimal solutions, $found(i)$ is the number of times the algorithm located the optimal solution i, and $rob(i)$ is the mutational robustness of the optimal solution i.

The default common parameters for each algorithm are given in Table 2. Several different values were tried for both the number of generations and the population size. The choice of population size did not have a significant impact and the number of generations was chosen so as to allow sufficient time for robustness to develop.

Parameter	Symbol	Value
No. Trials	t	100
No. Generations	g	200
Population Size	p	100
No. Mutations	m	1
Grid Width	x	3
Grid Height	y	3
Representation Length	l	10

Table 2: Default parameter values shared by both algorithms.

For the more difficult problem variants (4x4 grid with representation lengths of 16 or 17) both algorithms occasionally failed to find any optimal solutions in the allotted number of generations. These trials were not counted towards the 100 trials run for each experiment.

2.2.1 Stochastic Hill Climber

```
01. Initialize Array of Solutions
02. Do g Times
03.         Mutate Each Solution in the Array
04.         If the New Solution has a Fitness ≥ the Old
              Solution Save it
05.         Record the Fitness for Each Solution
```

Stochastic Hill Climbing Algorithm

The stochastic hill climber (HC) algorithm was implemented as an array of p stochastic hill climbers. Each individual hill climber was initialized with a distinct randomly determined set of instructions. Mutation was accomplished by randomly picking m instructions, without replacement, and replacing each one with a new directional instruction chosen uniformly at random from the four possibilities.

2.2.2 Tournament Selection EA

```
01. Initialize Population of Solutions
02. Do g Times
03.         Divide the Population Randomly into
              Tournaments of Size 4
04.         Copy the 2 Most Fit from Each Family into
              the 2 least fit, creating 2 Children
05.         Perform 1pt Crossover on the Children
06.         Mutate Each Child
07.         Record the Fitness for Each Solution
```

Tournament Selection Algorithm

The tournament selection (TS) algorithm used a tournament size of 4 and 1 point crossover. For each tournament the children of the best two individuals replaced the worst

Grid Size	Representation Length	No. Optimal Solutions	Fraction of the Solution Space	Minimum Robustness Value	Maximum Robustness Value
3x3	8	8	1.22e-4	0.25	0.25
3x3	9	136	5.19e-4	0.25	0.416667
3x3	10	1524	1.45e-3	0.25	0.525
3x3	11	13272	3.16e-3	0.25	0.613636
4x4	15	52	4.84e-8	0.25	0.25
4x4	16	1346	3.13e-7	0.25	0.359375
4x4	17	22130	1.29e-6	0.25	0.441176
4x4	18	272792	3.97e-6	0.25	0.5

Table 1: Basic properties for each of the variations of the SAW problem used in the experiments. These properties are: (1) the number of optimal solutions, (2) the fraction of the solution space occupied by the optimal solutions, (3) the minimum robustness value of any optimal solution, and (4) the maximum robustness value of any optimal solution.

two individuals every generation. The children underwent 1pt crossover with a probability of 0.8 and were always mutated. The mutation operator was implemented the same as in the HC algorithm. m instructions were replaced with new directions chosen from the four possibilities uniformly at random.

The two earlier studies demonstrated that the robustness effect occurred using the same tournament selection method, but without the crossover operator. The crossover operator was included in this study in order to test whether or not the results would hold up with a more "typical" implementation of a tournament selection based genetic algorithm.

2.3 Experimental Design

Three different sets of experiments were run and analyzed. Each set of experiments was designed to probe a different aspect of the algorithms and the SAW optimization problem in the hopes of illuminating the mechanism by which tournament selection finds significantly more robust optimal solutions. Significance was determined by comparing the means of the average robustness scores for each pair of algorithms using the Z test statistic.

2.3.1 Robustness Effect

The first set of experiments was designed to determine which of the two algorithms generated more robust optimal solutions: the stochastic hill climber or the tournament selection based genetic algorithm. The results for both algorithms were also compared with the baseline, determined through random sampling of the optimal solutions previously enumerated.

Both algorithms were run on six different variations of the SAW problem. The first three variants all used a 3x3 grid, but varied in their representation length. Since there was no variation in robustness for the set of minimally optimal walks only representation lengths greater than 8 were used; specifically: 9, 10, and 11.

The second set of three variants of the SAW problem all used a 4x4 grid. Again three different representation lengths were examined: 16, 17, and 18. The upper bounds of 11 and 18 on representation length as well as 4x4 on grid size were chosen to keep the number of optimal solutions manageable. To test larger parameter settings would require changing from a complete enumeration of optimal solutions and their mutants to a less exact sampling method to determine robustness.

2.3.2 Mutational Effect

In the two previous studies the mutation operator used in the search algorithms was limited to a single point mutation. In this set of experiments the effect that increasing the number of point mutations has on the robustness of the solutions obtained is examined. Five different values of m were tested: 1, 2, 3, 4, and 5.

As a coda to the experiments with the mutation operator two versions of the tournament selection algorithm were compared. The default implementation used the crossover operator with a probability of 0.8, while the other implementation did not use the crossover operator at all.

2.3.3 Evolutionary Time

The final set of experiments explored the robustness of the discovered optimal solutions as a function of the number of generations the algorithms were allowed to run. The search problem posed by the 3x3 grid with a representation length of 10 is simple. As shown in Table 1 optimal solutions make up $1.45 \cdot 10^{-3}$ of the total search space. This means that given a population size of 100 and 100 trials an optimal solution will appear in the initial population 14.3% of the time.

What remains unclear, however, is how soon after an optimal solution is located the algorithms begin finding robust solutions. Both algorithms were run for seven different generation times: 10, 25, 50, 100, 200, 400, and 800. The average robustness of the optimal solutions found was reported. For a single sample run the average robustness of the optimal solutions in the current population was calculated and reported for each of 500 generations.

The discovery of the first optimal solution is used as a milestone instead of convergence primarily due to the difficulty defining convergence in the context of the hill climber which does not have a pooled population.

3. RESULTS

3.1 Experiment 1: The Robustness Effect

The results for the three SAW variants run on a 3x3 grid are given in Figure 2. The figure shows that the optimal solutions located by the GA were significantly more robust than those obtained by either the random selection of optima or the hill climber. In fact, the optimal solutions obtained by the hill climber were significantly less robust than those

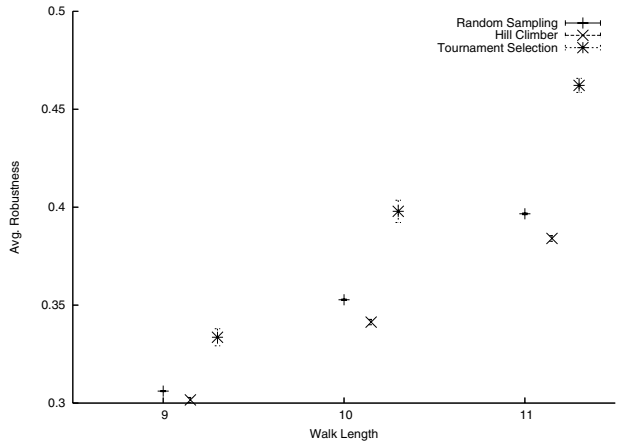

Figure 2: A plot of the average robustness of the solutions found by each algorithm with 95% confidence intervals as the algorithms are run for three distinct walk lengths(9, 10, and 11) on a 3x3 grid.

sampled at random from among the optimal solutions. This suggests that upon finding the edge of the optimal solution space the hill climbing algorithm did not stray very far from it.

As the length of the representation was increased the gap between the methods also increased. One possible reason for this could be that the range in robustness available is also increasing. This effect is explored further in the conclusions.

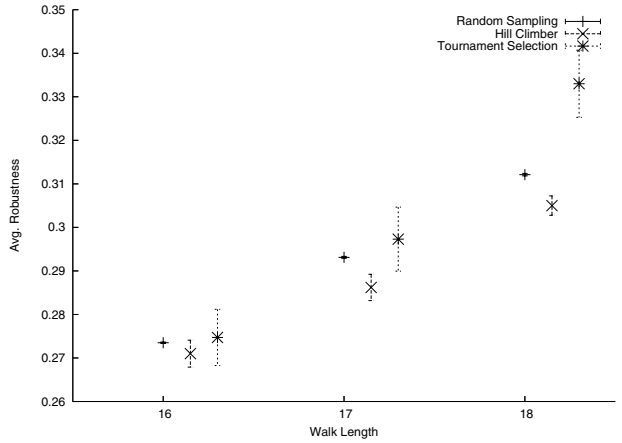

Figure 3: A plot of the average robustness of the solutions found by each algorithm with 95% confidence intervals as the algorithms are run for three distinct walk lengths(16, 17, and 18) on a 4x4 grid.

The SAW optimization problem posed by the 4x4 grid is much more difficult, and the results reflected this. As can be seen in Figure 3 the ability of the genetic algorithm to find robust optimal solutions is highly dependent on the representation length. Only with a representation of length 18 did the genetic algorithm locate optimal solutions which were significantly more robust than those chosen uniformly at random. The hill climber, on the other hand, located optimal solutions which were significantly less robust than those located at random for representation lengths of both 17 and 18.

3.2 Experiment 2: The Mutational Effect

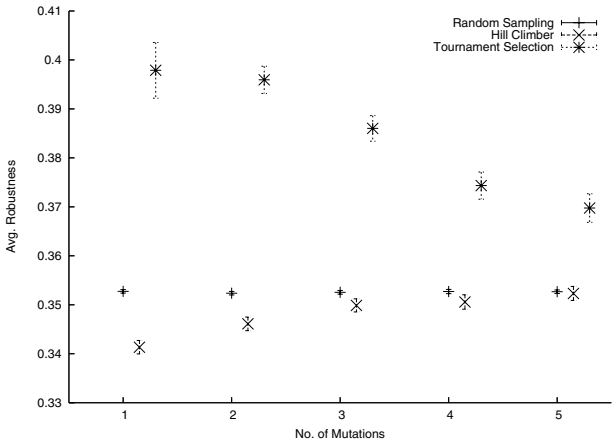

Figure 4: A plot of the average robustness of the solutions found by each algorithm with 95% confidence intervals as the algorithms are run for five different mutation settings: 1,2,3,4, and 5.

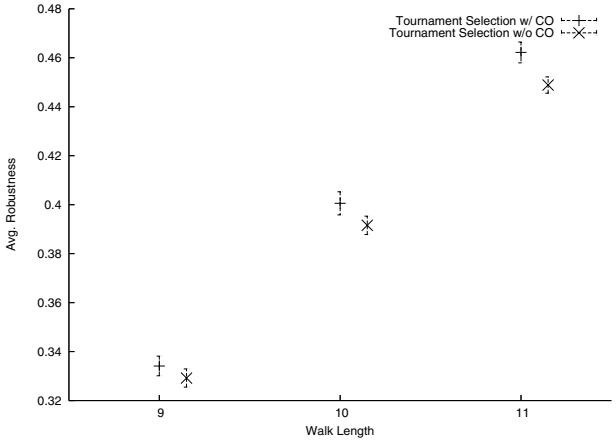

Figure 5: A plot showing the effect of the crossover operator on the average robustness of the solutions found by the tournament selection algorithm with 95% confidence intervals.

Increasing the number of point mutations applied each generation had quite a pronounced effect on both of the algorithms. As can be seen in Figure 4 increasing the frequency with which the mutation operator was applied had a significantly disruptive effect on the robustness of the solutions obtained by the GA. At the same time, the HC algorithm benefited greatly from increasing m. For values of 4 and 5 the HC algorithm no longer found optimal solutions which were significantly less robust than those chosen at random. This suggests that a higher mutation rate helps drive the HC algorithm into randomly exploring the optimal solution space. In spite of the effects of increasing the frequency of mutation the genetic algorithm still found optimal solutions which were significantly more robust than those found by the other methods.

The crossover operator, on the other hand, not only helped the algorithm find solutions more quickly, but in one case even helped find significantly more robust solutions. This behavior is shown clearly in Figure 5. For representation lengths of 9 and 10 the solutions obtained by the implementation of the algorithm with crossover were not significantly more or less robust than those obtained by the implementation without crossover. For a representation length of 11, however, the implementation with crossover actually found slightly more robust solutions. The implementation with crossover found an optimal solution, on average, by generation 3. The implementation without crossover found optimal solutions, on average, by generation 4.

3.3 Experiment 3: Emergence of Robustness

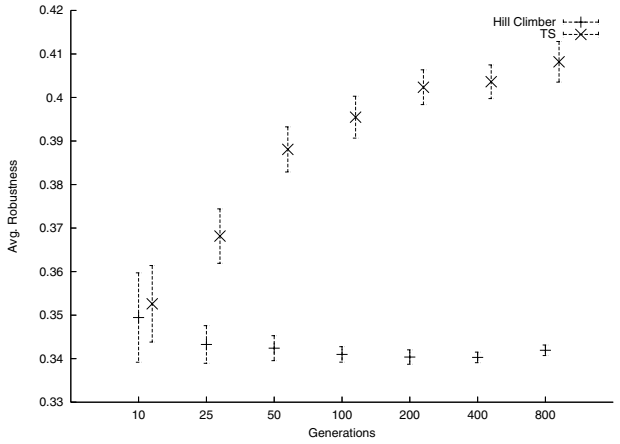

Figure 6: A plot of the average robustness of the optimal solutions found by each algorithm with 95% confidence intervals as the algorithms are run for a successively larger number of generations.

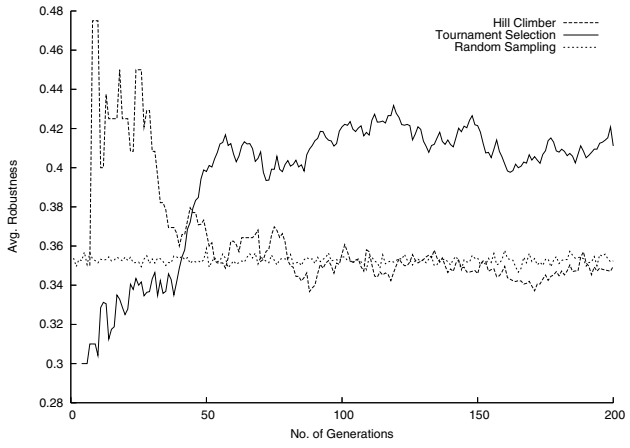

Figure 7: A plot of the average robustness of the optimal solutions in the current population versus time for both algorithms and a random sampling of the optimal solutions.

The effect of allowing the algorithms to run for longer and longer numbers of generations is shown in Figure 6. At a generation time of 10, shortly after an optimal solution was found by both algorithms the average robustnesses of the populations was not significantly different. After only 15 additional generations, however, the population of optimal solutions found by the tournament selection algorithm was significantly more robust. The gap between the two populations only grew as the generation time increased.

A typical sample run for each algorithm showing how the robustness changed over time is given in Figure 7. At each generation the average robustness of the current group of optimal solutions was reported. If there were no optimal solutions then the point was not plotted. On average the hill climber found the an optimal solution by generation 5 and the tournament selection algorithm by generation 3. Both algorithms reached an equilibrium state around generation 100. The tournament selection algorithm stabilized well above the randomly sampled optimal solutions. The hill climber, however, stabilized just below the random solutions.

4. CONCLUSIONS

The results of the first experiment suggest that while the tournament selection algorithm tends to find significantly more robust solutions than either chance or the hill climber, it does not always do so. In order to further investigate the histograms of the robustness values of all optimal solutions for each of the SAW variants were examined. The histograms for the 3x3 variants are given in Figure 8 and for the 4x4 variants in Figure 9.

As can be seen in the table the robustness values for the 3x3 variants form a symmetric bell curve, while the 4x4 variants form a highly skewed distribution. In work not shown here it was verified that as the grid size increases from 2x2 to 5x5, the distribution of the variances becomes more and more skewed. The correlation between the skew and completeness of the distributions of the robustness values and the ability of the genetic algorithm to find robust solutions suggests that the effectiveness of genetic algorithms at finding robust solutions is highly dependent on the nature of the optimal solution space.

The second experiment showed that the crossover operator, for the SAW problem at least, was neither a major cause nor a preventer of robustness. While the 1pt crossover used here did tend to have a slightly beneficial effect on the development of robustness this effect did not appear to be the major reason why the GA was so successful at locating mutationally robust optimal solutions. Although it is not explicitly shown in Figure 5 the average robustness scores of the GA run without crossover were still significantly higher than those obtained from either the stochastic hill climber or random sampling. This suggests that the only thing separating the hill climber from the tournament selection algorithm is the selection method and the interconnected nature of the population in the GA. This result also implies that the development of robustness is primarily a function of the mutation operator, although the results for representation length 11 suggest that the crossover also plays a role.

This leads us to suggest that in the GA robust solutions act as weak attractors. A robust solution, by definition, is more likely to have optimally fit children than a non robust solution. Therefore, the children of robust solutions are more likely to survive and have children of their own. The second set of experiments also demonstrates that too much mutation is inhibitive to the development of robustness.

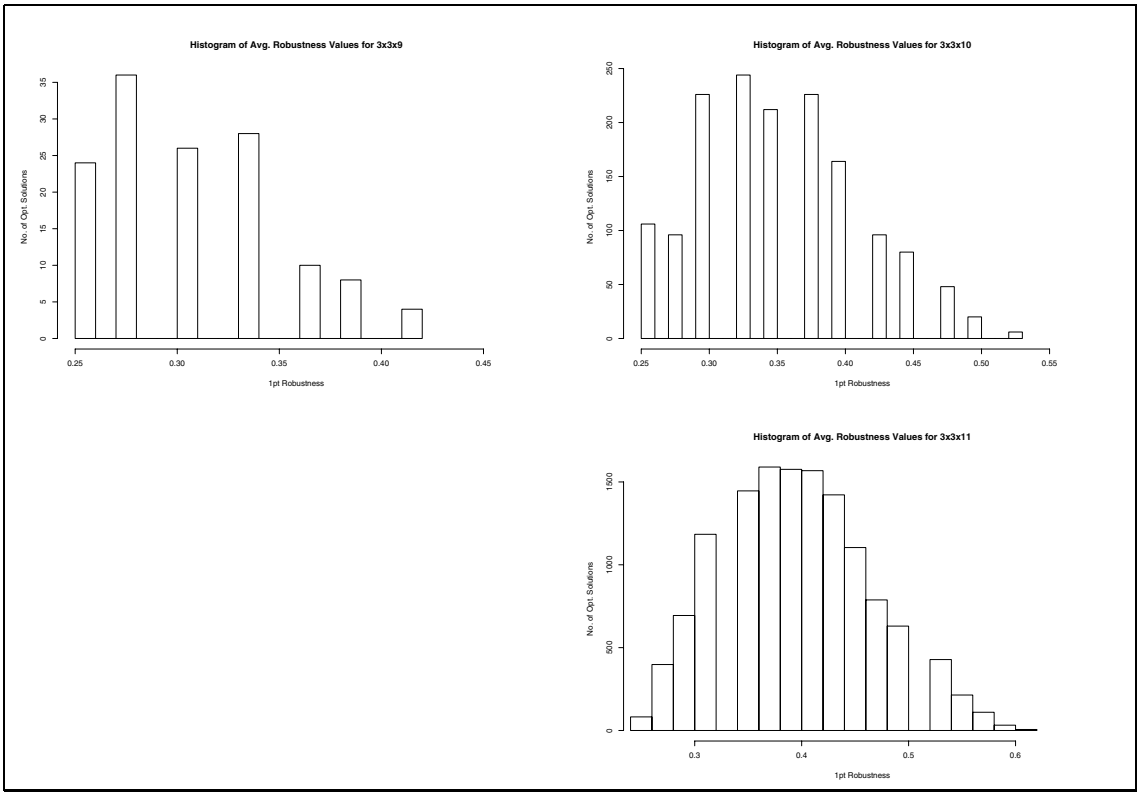

Figure 8: Histograms of the mutational robustness for every optimal solution to each of the three 3x3 grid variants. Beginning in the upper left and proceeding clockwise the variants have representation lengths of 9, 10, and 11 respectively.

One potential explanation for the lack of robustness in the low representation length 4x4 grid variant is simply that you need a certain level of available robustness before the robust solutions can act as effective attractors.

The third grouping of experiments is perhaps the most interesting. The TS algorithm starts to find robust solutions quite quickly. This suggests that it may be relatively cheap in computational cost to let a simulation run beyond the location of a satisfactory solution in order to generate a higher robustness within that population.

5. FUTURE DIRECTIONS

This study suggest two main avenues of research. The first is in the area of real world applications. As was stated in the introduction genetic algorithms have been used to find solutions for a number of different engineering problems. It would be incredibly useful to the engineering community if it could be demonstrated that genetic algorithms consistently find more robust solutions simply by running for a small number of additional generations past the point at which a sufficiently optimal solution is found. Although the ability of genetic algorithms to find more robust solutions has now been demonstrated for several different problems a comprehensive study of real world engineering problems is needed to determine the practical benefits of this effect.

The second main direction suggested by this research is a further investigation of the theory behind the development of robust solutions by GAs. A number of unanswered questions still remain: How much of a role does the selection method play? Would a different GA, such as a fitness proportionate GA, find equally robust solutions?

It seems fairly clear from this research that the method of selection used plays an important role in the development of robustness. Studying graph-based genetic algorithms [2, 4], alternative means of selection, or other methods which control the interaction between members of the population pool could provide additional insight.

6. ACKNOWLEDGMENTS

The author would like to acknowledge Dr. Daniel Ashlock and Dr. Sushil Louis for many helpful discussions. This material is based upon work supported by the National Science Foundation under Grant No. 0447416.

7. REFERENCES

[1] T. Back, D. B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. Institute of Physics Publishing, 1997.

[2] K. M. Bryden, D. Ashlock, S. Corns, and S. Willson. Graph based evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 2006.

[3] S. Bullock. Will selection for mutational robustness significantly retard evolutionary innovation on neutral networks? In Standish, Abbass, and Bedau, editors, *Artificial Life VIII*, pages 192–201. MIT Press, 2002.

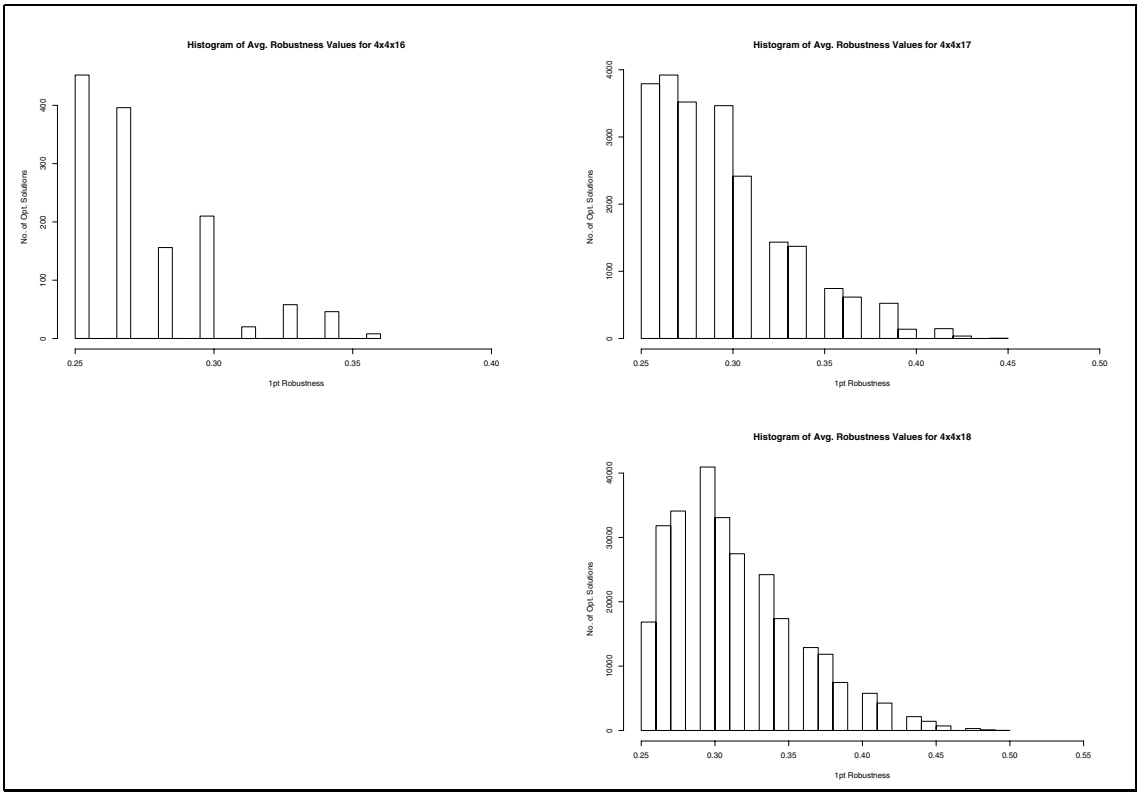

Figure 9: Histograms of the mutational robustness for every optimal solution to each of the three 4x4 grid variants. Beginning in the upper left and proceeding clockwise the variants have representation lengths of 16, 17, and 18 respectively.

[4] S. Corns, D. Ashlock, D. McCorkle, and K. Bryden. Improving design diversity using graph based evolutionary algorithms. In *Proceedings of the 2006 Congress on Evolutionary Computation*, pages 1037–1043. IEEE Press, 2006.

[5] J. Edwards and B. O. Palsson. Robustness analysis of the *escherichia coli* metabolic network. *Biotechnology Progress*, 16:927–939, 2000.

[6] D. Keymeulen, R. Zebulum, Y. Jin, and A. Stoica. Fault-tolerant evolvable hardware using field-programmable transistor arrays. *IEEE Transactions on Reliability*, 49:305–316, 2000.

[7] M. A. Marra, B. E. Boling, and B. L. Walcott. Stability analysis of genetic algorithm controllers. In *Conference Proceedings of IEEE Southeastcon 96*, volume 1. IEEE Press, 1996.

[8] J. Schonfeld and D. Ashlock. Comparison of robustness of solutions located by evolutionary computation and other search algorithms. In *Proceedings of the 2004 Congress on Evolutionary Computation*, volume 1, pages 250–257. IEEE Press, 2004.

[9] J. Schonfeld and D. Ashlock. A study of evolutionary robustness in stochastically tiled polyominos. In *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, volume 1, pages 19–26, 2005.

[10] D. Taverna and R. A. Goldstein. The distribution of structures in evolving protein populations. *Biopolymers*, 53:1–8, 2000.

[11] G. L. Urban, K. M. Bryden, and D. A. Ashlock. Engineering optimization of an improved plancha stove. *Energy for Sustainable Development*, 6(2):9–19, 2001.

[12] E. van Nimwegen, J. P. Crutchfeld, and M. Huynen. Neutral evolution of mutational robustness. In *Proceedings of the National Academy of Sciences of the United States of America 1999*, volume 96, pages 9716–9720, 1999.

[13] A. Wagner. *Robustness and Evolvability in Living Systems:(Princeton Studies in Complexity)*. Princeton University Press, 2005.

[14] A. Wagner. Robustness, neutrality, and evolvability. *FEBS Letters*, 579:1772–1778, 2005.

[15] C. O. Wilke, J. L. Wang, C. Ofria, R. E. Lenski, and C. Adami. Evolution of digital organisms at high mutation rate leads to survival of the flattest. *Nature*, 412:331–333, 2001.