Implementation of a Genetic Algorithm to Simulate the Evolution of CoreWar Warriors Based on Fitness Values

MOHAMMAD ASHRAF SIDDIQUEE, University of New Mexico
VANESSA SURJADIDJAJA, University of New Mexico

Genetic algorithms (GAs) simulate the evolution of a population within a reasonable amount of time. GAs offering us the ability to predict the long-term effects on a population based on the various circumstances we initialize at the start of the GA. This paper looks at the fundamentals of constructing a GA and how altering various conditions, such as selection operators, crossover, and mutation rates, affect a population over multiple generations. After varying the previously mentioned variables within our GA, we found that, given a significant amount of time, a population will eventual converge to genomes that perform in a similar manner. Thus, the population stops exhibiting major alterations to its genomes despite the presence of selection, crossover, and mutation operators affecting individuals of the population at each generation. The convergence we observed over the population's evolution and the incomparable genes of the top individual over three trial-runs suggest that the elitism applied to our GA converges our warriors towards a genome that is proven to be successful in previous generations but can also adapt through mutation.

Additional Key Words and Phrases: Genetic algorithms, evolution, biological simulation, selection, crossover, mutation

ACM Reference format:
Mohammad Ashraf Siddiquee and VANESSA SURJADIDJAJA. 2016. Implementation of a Genetic Algorithm to Simulate the Evolution of CoreWar Warriors Based on Fitness Values. 1, 1, Article 1 (January 2016), 7 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

AUTHORS CONTRIBUTION

Author1 wrote the final skeleton script of the genetic algorithm used in this experiment, wrote the final code for the genetic algorithm, analyzed the output of warriors when initial conditions were altered, and ran the genetic algorithm to obtain two best warriors.

Author2 wrote the initial skeleton script of the genetic algorithm, wrote the code for the genetic algorithm's mutation method, wrote the code for implementing an Island GA, debugged for the program, tested the various conditions discussed in the paper, collected and analyzed results from trial-runs of the GA, created the figures using Matlab, maintained the GitHub repository, and wrote/maintained the Overleaf document.

1 INTRODUCTION

The adaptability of an individual within a system can be associated with its genome, despite how rudimentary an individual's genome may seem. As Dario Floreano and Laurent Keller explains, genes do not directly affect an individual's behavior, rather they "encode molecular products that lead to the development of brains and bodies through which behavior is expressed"[3]. The study of genomic evolution allows us to further understand the long-term impacts of an individual's surroundings. However, to observe the long-term effects of genomic evolution, the implementation of genetic algorithms (GAs) provides researchers with the flexibility to test various conditions within a reasonable amount of time.

In this paper, we explore the effects of applying a GA to a population of individuals. By experimenting with various selection, mutation and crossover operators, we find that selection and crossover operators with a low crossover rate can be used to preserve warriors with successful gene combinations. This form of preservation allows the population to explore and expand their fitness landscape without the risk of losing integral parts of its genome.

2 RELATED WORKS

GAs have been frequently used to evolve programs in response to the difficulties these programs encounter while executed. Floreano and Keller focus on the adaptive behavior of robots and how these robots utilize Darwinian selection in their GAs. Floreano and Keller duly note that the complexity of these evolving robots are not expressed with complex, fundamental components but rather utilize simple components, concatenated in various manners, to create complexity [3]. While GAs are complex and hierarchical as a whole, the individual parts of a GA are relatively simple in their functions.

What we program and observe in GAs are not necessarily unique. Many of the guiding principles towards constructing GAs draw from research on gene networking observed in the biological sciences. As Ben-Jacob highlights, bacteria create complex colonies despite their inability to store all the required information in their genes pertaining to the creation of such infrastructure [2]. Ben-Jacob continues by suggesting that the information gained from the environment can be condensed within the organism and used as a guiding description for the organism's function.

In their analysis of GAs, Suzuki and Iwasa observed the fibabel-like,fi or plateau, fitness function in GAs, where a single individual is favored and preserved among the rest of the population. However, by suggesting that crossover results in a plateau fitness function, Suzuki and Iwasa's GA experienced an exponential increase in their population's fitness only to have the population reach a maximum threshold for their fitness before having the fitness decrease exponentially [7].

For Smith et al., GAs that experience some form of elitism through intense competition result in a fiflat fitness landscape,fi which illustrates the allele's versatility in the genome. Smith observed the versatility of the gene segment Env C2-V4 in HIV-1. Smith found that the C2-V4 alleles in HIV-1 exhibits a fiflat fitness landscape,fi which Smith defines as the allele's bias toward their fitness. Furthermore, Smith suggests that such a fitness landscape illustrates "the

DOI: 10.1145/nnnnnnn.nnnnnnn

capacity for subtle polymorphisms in Env to nevertheless significantly impact viral fitness" [6]. It is the polymorphism of the allele segment that allows HIV-1 to easily adapt to its surroundings.

3 METHODS

To obtain a testable output for our GA, we used Redcode, an assembly language, as the genes in an individual's genome within the population. The Redcode that was generated were then manipulated through crossover and mutation in our GA. We created a GA that contained the following classes: Gene, Warrior, Population, Utilities, and Definitions. Gene contained the command, parameters, and address modifiers that make-up a single Redcode instruction.

The object Warrior contains an ArrayList of genes, making up an individual warrior's genome. When these warriors are entered into the Redcode program pmars, their assembly language compete with one another and a benchmark warrior to remove one another out of the CPU's memory. Once the program is completed, each warrior is given a fitness score that is calculated from the number of wins and ties each warrior had during the run of the program.

The object Population is an ArrayList of warriors where each time a selection operator is called within the main program, the list of warriors within the population are sorted with respects to their fitness score. The Utilities and Definitions classes contain package-shared final variables and functions, such as a probability function, and the constants used to signify which type of selection and crossover operators are being implemented. The Utilities class also holds the crossover and mutations rates called for throughout the program.

Once the GA was completed, we varied the selection method from Roulette [1], Tournament [2], and Random [3] while maintaining crossover as Uniform with a rate of 0.5 and the mutation rate at 0.5. We then maintained the previously mentioned specifications as our selection method while varying our crossover methods from no crossover [4], One-point[5], and Uniform[6]. Mutation rate was kept at 0.5. Once we obtained the results for varying selection and crossover operators, we then varied the crossover rates from 0.2, 0.5 and 0.8 and then varied the mutation rate in a similar manner by using the rates 0.2, 0.5, and 0.8.

Since our GA utilized elitism throughout the program, it was necessary to implement an island GA to evolve our population out of the local maxima the population was biased towards. The island GA we implemented took the top three warriors from the population and had them reproduce and evolve separately (i.e. populate and evolve on separate fiislandsfi). At the end of 20 generations, we then re-introduced the best warriors from the three populations into the initial population and generate another 20 generations with the addition of the newly evolved warriors.

We also tested the effects of elitism on our GA by implementing code that would eliminate the elitism biases programmed into the GA. Elitism biases were eliminated by adding a random warrior to the current generation from the previous generation. Our GA terminates based on the number of generations indicated from the Definitions class in the program. For the purposes of this experiment we terminated the GA after 20 generations.

4 RESULTS

The following results correspond to the varying of the selection operators:

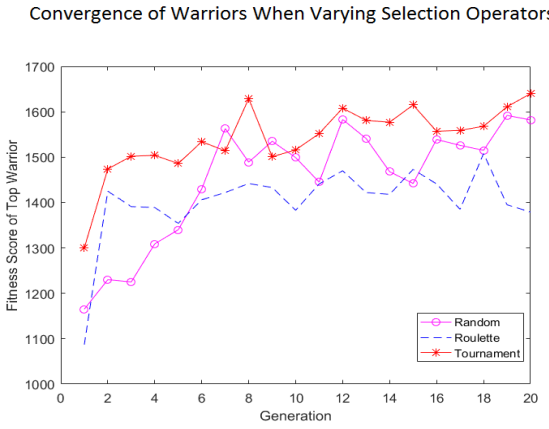

Fig. 1a. The convergence of the top warriors from each generation when varying the selection Operators in the GA.

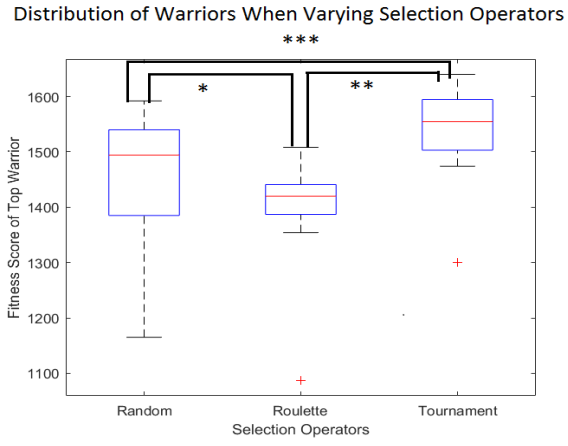

Fig. 1b. The distribution of warriors throughout the final population with respects to the selection operator used in the GA.

As seen in Fig. 1a, after varying the selection operators in our GA, we found that over 20 generations, the population begins to converge around generation 6. While the fitness score of the top warrior

[1]Roulette calculates the proportion of warriors with a certain fitness score and inserts that proportion of warriors into the next generation.
[2]Tournament takes the warrior with the highest fitness score and adds the warrior into the next generation.
[3]Random will take a random warrior from the top half of the population, after the population has been sorted based on the individual warrior's fitness score, and replace the lower half with one of these warriors.
[4]In order to simulate this condition, crossover rate was reduced to 0
[5]One-point finds a random segment of varying length in the warrior's genome and swaps it with the segment of varying length of another warrior's genome.
[6]Uniform crossover will swap each gene if, and only if, the probability of the crossover exceeds the indicated value in the Definitions class.

of each generation slightly fluctuates, their fitness scores remain within the same range. With respects to our GA, a Tournament style selection operator yielded warriors with higher fitness scores over the course of 20 generations. The distribution of the warriors, illustrated in Fig 1b, were more dispersed in Random style selection, while Tournament style consistently had warriors with fitness scores in the range of 1500 to 1600. However, Roulette style exhibited a more consistent, stable population with regards to their fitness score. The p-values are as follows: *p-value = 0.0200, **p-value = 2.0571e-06, and ***p-value =0.0200.

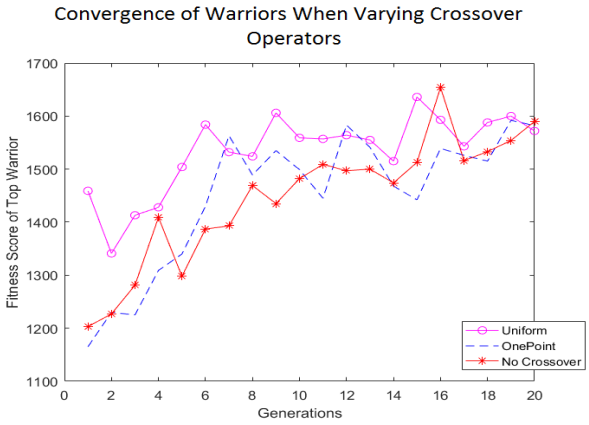

Fig. 2a. The graph plots the convergence of the top warriors in each generation when varying the crossover operators.

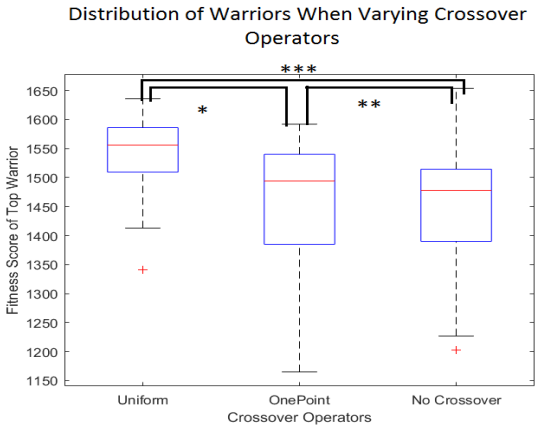

Fig. 2b. The plot illustrates the distribution of the warriors when varying the crossover operators.

As illustrated in Fig. 2a, there is an initial difference in the populations when varying the crossover operators. However, this difference existed until generation 10. From Generation 11 to 20, the populations began to converge to one fixed range of fitness scores. However, given 20 generations, all three crossover operators converged on a similar range of fitness scores, 1500 to 1600. Fig. 2b demonstrates the distribution of the warriors in the last generation of each crossover operator tested. The p-values were *p-value = 0.0200, **p-value = 0.5979, ***p-value = 0.0200. Uniform crossover exhibited a more concise distribution of warriors that had a fitness score range between 1500 and 1600. One-point crossover presented a larger distribution of fitness scores, while implementing no crossover led to a distribution in-between Uniform and One-point crossover.

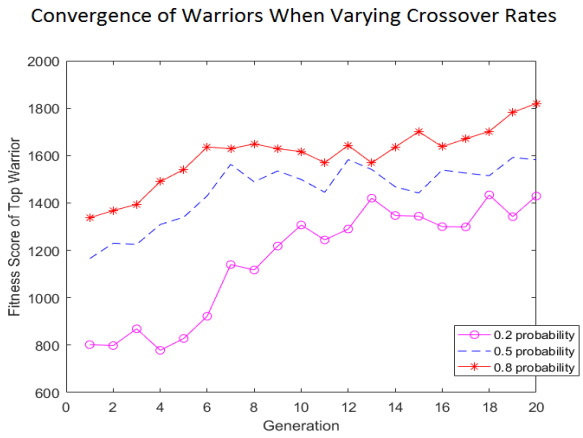

Fig. 3a. The convergence of the top warriors from each generation when varying the selection Operators in the GA.

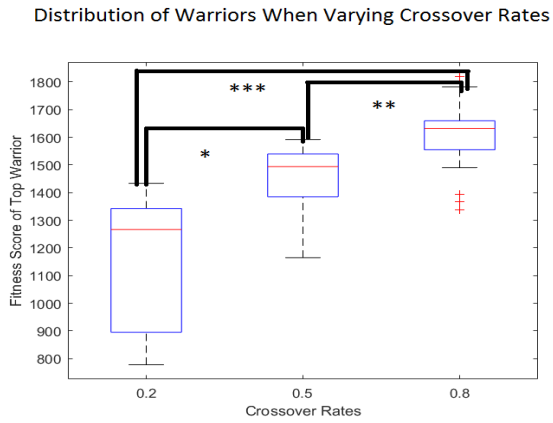

Fig. 3b. The distribution of warriors throughout the final population with respects to the selection operator used in the GA.

When varying the rate of crossover in our GA, the three different rates chosen, 0.2, 0.5, and 0.8, presented us with warriors with fitness scores proportional to the rate indicated at the start of the GA. As Fig. 3a illustrates, a crossover rate of 0.8 yields warriors with fitness scores that fluctuated between 1600 and 1800 from around generation 10 to generation 20. However, all crossover rates began to converge in earlier generations, around generation 7. Fig. 3b demonstrates the distribution of the warriors in generation 20, where a crossover rate of 0.2 resulted in a distribution that spanned

fitness scores of 800 to 1300. However, an increase of the crossover rate to 0.8 resulted in a distribution of warriors with a smaller range of fitness rates. The p-values were *p-value = 3.7051e-05, **p-value = 3.2047e-04, ***p-value = 3.7051e-05.

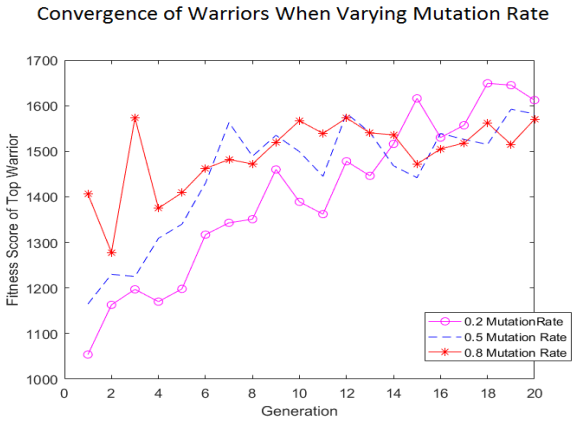

Fig. 4a. The convergence of the top warriors from each generation when varying the selection Operators in the GA.

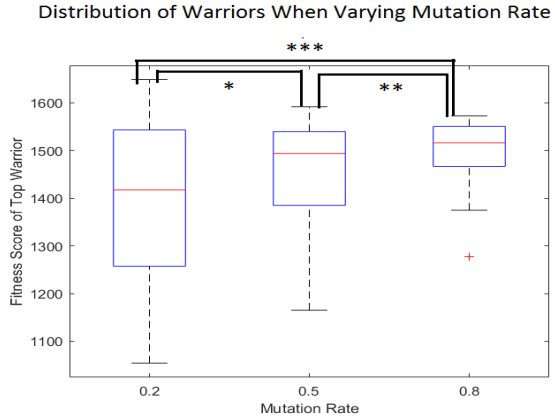

Fig. 4b. The distribution of warriors throughout the final population with respects to the selection operator used in the GA.

Fig. 4a illustrates that, over time, a mutation rate of 0.2, 0.5, and 0.8 will result in a convergence of the population in later generations. We begin to see signs of convergence towards generation 10 for mutations rates of 0.5 and 0.8. A mutation rate of 0.2, on the other hand, experienced a gradual increase in the top warrior's fitness score. However, signs of convergence appeared around generation 18. The distribution of warriors, as seen in Fig. 4b, when varying the mutation rate of our GA is much greater for a mutation rate of 0.2 compared to a mutation rate of 0.8.The p-values were *p-value = 0.4735, **p-value = 0.4734, ***p-value = 0.4735.

If we take into consideration the distribution of warriors in each varied initial condition, we see that the fitness landscape explored by our warriors in each run of the GA usually exists between fitness scores of 1300 and 1600. However, a mutation rate of 0.2 was the only varied initial condition that led to warriors exploring genomes that yielded fitness scores lower than the previously mentioned range. Due to elitism within our GA, the fitness landscape is simple since all warriors are encouraged to evolve towards the higher fitness score preserved from previous generations.

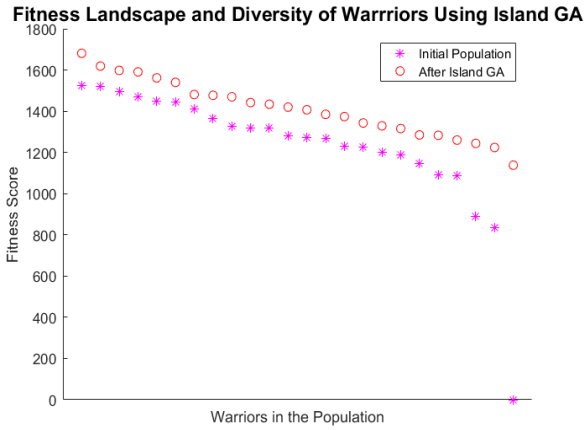

Fig. 5. The distribution of each warrior's fitness score in the initial population and the population after implementing Island GA.

With respects to our Island GA, the fitness scores were equally distributed from fitness scores ranging from 1100 to 1600. The implementation of an Island GA allowed the population to continue exploring a fitness landscape that differed from the fitness landscape of the initial population. As seen in Fig. 5, after implementing the Island GA to our original GA, the population was able to maximize its overall fitness score by 200 points

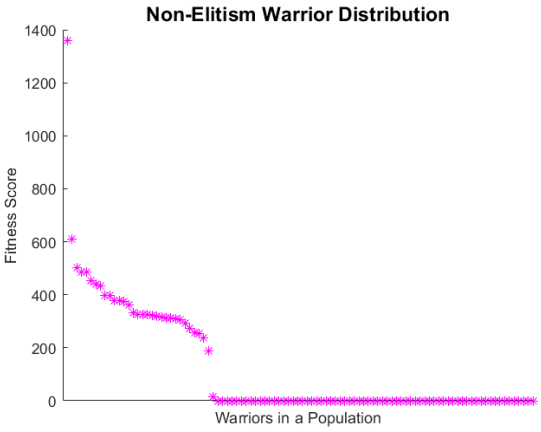

Fig. 6. The distribution of each warrior's fitness score in the population after implementing non-elitism.

As Fig. 6 illustrates, the implementation of non-elitism into our GA resulted in lower fitness scores for warriors of generation 20. Most warriors exhibited fitness scores of zero while one warrior

had a score in the range of 1400. Nevertheless, the warrior with the highest fitness score still has a score that was below average when compared to a warrior bred through elitism.

5 CONCLUSIONS

Our GA optimally runs with the initial conditions set to Tournament selection with Uniform crossover, a crossover rate of 0.8, and a mutation rate of 0.2. In addition to elitism, the use of Tournament selection and Uniform crossover further preserves the best warriors from each population. Tournament selection preserved the best warrior by choosing the warrior with the highest fitness score. Uniform crossover at a 0.2 crossover rate, preserves the best warriors even further by decreasing the chances a warrior will lose vital segments of their genome that contribute to a high fitness score.

A mutation rate of 0.8, on the other hand, allowed any randomly generated population to eventually create a warrior with a genome that yields a high fitness score over the course of at least 20 generations. In other words, a high mutation rate leads to the population's expansion of its fitness landscape while the preservation effects of Tournament selection and Uniform crossover at a rate of 0.2 tethers the population to a set of warriors whose genome comprise of genes that have been tested to be successful. Furthermore, a crossover rate of 0.2 contributed to the preservation of the best warriors due to the epitasis of the warriors' genes.

Arjan et al. defines epitasis as the interaction of various genotypes and alleles to create a phenotype in an individual. Rather than the notion of additive genetics, epitasis looks at interconnected genotypes and how alleles suppress or enhance the performance of other alleles[1]. If we look at the Redcode of our warriors, we see that one instruction set can depend on the previous and subsequent instruction sets. Sign epitasis, as Gerhard Schlosser defines the term, is the dependence of two different loci in order to push towards the fitness peak of the population [5]. For the purposes of our experiment, a lower crossover rate aided the process of epitasis in our populations. Warriors who preserve gene segments that resulted in high fitness scores from the previous generation will receive a higher fitness score than those who either performed crossover that separated the two genes or mutated either one, if not both, of the genes.

Testing our GA with non-elitism resulted in an immense decrease in fitness score for all individuals in the population even when the number of generations was increased to 100. Our reasoning for increasing the number of generations the population breeds in our GA was to allow our GA with non-elitism to have some means of exploring a fitness landscape that is relatively equivalent to our original GA with elitism.

However, despite the attempt to equalize the fitness landscape, the loss of the best warrior at each generation means that the fitness landscape the population explored during that generation resets with the creation of subsequent generations. Therefore, at each generation warriors are susceptible to exploring gene combinations that previous warriors proved to be either figenetic baggagefi or, worse, fatal.

One way in which our GA can diversify its genome alongside its use of elitism is to incorporate an Island GA. By taking the best three warriors of the population[7] and allowing them to generate their own population consisting of warriors with the same genome. An interesting result from the test was that the final population consisting of the top warriors from each island population and the initial population saw an overall increase in each warriorfis fitness score by generation 20. However, the top warrior's fitness score maxed around 1600 which is considerably low compared to other trials that did not use an Island GA.

Studying the effects of inbreeding and interbreeding among Darwinfis finches, Grant et al. found that inbreeding of the species lead to the decrease of the fitness score of the species. When these species were interbred with one another, the fitness score of the next generation was decreased further compared to the control of finches not inbred nor interbred and the finches who were inbred [4]. Grant et al.'s results mimicked our warriors when they inbred in the island populations and interbred in the final population with one another and the remaining individuals not chosen to migrate to the island populations from the original population.

Inbreeding and interbreeding combined results in lower fitness scores due to an isolation of the population to a limited fitness landscape and a bias towards choosing warriors with rigid genomes accommodating towards an island's specific conditions. When they are incorporated back into the population, there is a high chance that the fitness landscape explored by the warriors from the island populations are too specific and rigid to the islands and, therefore, may not be beneficial to the conditions of the initial populations they evolved from.

Based on the results of this experiment, we found that the complexity of GAs stem from the interaction of their simpler components. While the various components in the code of our GA merely stored, modified, and updated information stored in memory, embedding these instructions within one another and using these instructions to alter the genes, the warriors, and the population at each generation leads to the complexity of any GA. The next step for our experiment on GAs is to observe its performance alongside other GAs, further entwining our understanding of GAs and their complexity.

REFERENCES

[1] J. Arjan, G.M. de Visser, Time F. Cooper, and Santiago F. Elena. The causes of epistasis. *Proceedings: Biological Sciences*, 278(1725):3617–3624, Dec 2011. doi: 10.1098/rspb.2011.1537.

[2] Eshel Ben-Jacob. Learning from bacteria about natural information processing. *Natural Genetic Engineering and Natural Genome Editing*, 1178:78–90, 2009. doi: 10.1111/j.1749-6632.2009.05022.x.

[3] Dario Floreano and Laurent Keller. Evolution of adaptive behaviour in robots by means of darwinian selection. *PLoS Biology*, 8(8):1–8, January 2010. doi: 10.1371/journal.pbio.1000292.

[4] Peter R. Grant, B. Rosemary Grant, Lukas F. Keller, Jeffrey A. Markert, and Kenneth Petern. Inbreeding and interbreeding in Darwin's finches. *Evolution*, 57(12):2911–2916, Dec 2003.

[5] Gerhard Schlosser. Epistasis, constraints, and coevolution. *Evolution and Development*, 11(5):459–461, 2009. doi: 10.1111/j.1525-142X.2009.00353.x.

[6] S. Abigail Smith, Charles Wood, and John T. West. HIV-1 Env C2-V4 diversification in a slow-progressor infant reveals a flat but rugged fitness landscape. *PLoS One*, 8 (4):1–18, Apr 2013. doi: 10.1371/journal.pone.0063094.

[7] Hideaki Suzuki and Yoh Iwasa. Crossover accelerates evolution in GAs with a babel-like fitness landscape: Mathematical analyses. *Evolutionary Computation*, 7 (3):275–310, 1999.

[7] The best three warriors were chosen here since convergence of the warriors' fitness scores occurred with the top warriors in the population over the course of 20 generations

APPENDIX

The following are the two sample Redcode warriors generated from our GA:

```
; redcode
; name  Warrior_T9_001
; author  GROUP_T9
; assert    CORESIZE == 8000 && MAXLENGTH >= 100
JMZ $2, }4
SPL  4, }6
SUB $3, $5
DIV *0,  3
NOP *4, #7
JMN #0, <4
SNE }4, <6
LDP {4, *6
SEQ @1, #7
SPL  0, {2
SLT @2, $6
MOV *6, }7
NOP  6, }7
SEQ <3, }4
MOV {5, <7
SPL *4, @7
SPL {1, #6
SLT *3, }6
DAT {0, #5
MOD @5,  7
SPL #1,  4
JMP {2, }7
JMP {6, <7
JMN $3, {4
SEQ #3, *6
JMP @1, @7
MUL *4,  6
STP <5, #7
SNE *2, {7
DJN @1, $6
DIV {5,  6
DIV *2, $4
DAT  3, <6
DIV  2, $6
SEQ *2, <5
DIV <3,  6
SLT $1, *3
SEQ #1, #3
CMP }0, $5
DAT #6, $7
MUL @3,  6
JMN {2, }7
STP  0, $7
SEQ #2, #3
CMP }4, $7
SLT {5, *6
MUL }0,  3
JMZ  3, }7
ADD <0, $7
MOV {1, {2
STP {4, #5
DIV  1, }3
LDP #1, *7
SPL @1,  4
MUL *0, $5
JMP $6, {7
SPL $3, <7
JMP @3, <6
JMZ }4,  5
DJN @2, @4
JMP {3,  6
ADD #6, }7
DAT {5,  6
JMP <5, }6
SLT {4, @7
SLT {1, <3
SPL <1, }4
SUB  1, <4
SLT $0, @2
DIV *4, @7
ADD @1, <6
MUL @3,  6
SNE <6, @7
SLT @5, $7
SUB {0, #6
SNE *4,  6
DAT $3, *7
MOD $4, {6
MOV {4, @7
SPL {2, }7
SLT <6, *7
SUB $5, }6
MUL {4, @6
SPL <2, @3
JMP @0,  6
SUB {3, }7
MUL  0, *2
SLT <6, *7
SNE *2,  6
ADD <3, #6
SNE @3, <6
STP $5, <6
ADD <4, @7
JMZ }6,  7
SEQ {3, #7
MOV {1, @7
DAT {3, @6
DAT }0, {2
SLT #0, {2
JMZ *2, }5
end ;  execution  ends  here
```

```
; redcode
; name  Warrior_T9_002
; author  GROUP_T9
; assert    CORESIZE == 8000 && MAXLENGTH >= 100
JMN {5, *6
SUB {6, {7
DAT }4, <5
JMN <3, $7
ADD $0, *4
LDP *6, @7
MUL }6, *7
CMP *6, }7
LDP <1, *6
LDP #5, #6
SPL $1, <2
MOD }6, $7
LDP *5, #6
SUB <0, #5
SEQ #1, *2
SPL $6, }7
SEQ @0, #5
SPL #2, #3
LDP  1,  3
JMP #5, }6
JMN {4, *7
ADD #2, }4
SPL @0,  1
SPL $3,  5
SNE {3, @4
MUL $4, <5
MOD @4, }5
MOV *6, #7
SPL #0, $4
DJN }6, }7
DAT <1, #7
MUL <6, #7
STP @6, *7
MOD  2, #7
LDP #0, $1
JMN }3,  7
MUL *2, <5
DIV }6, #7
JMN {2, <5
DIV <2, }6
SNE #5, }6
MOV $6, #7
SUB $0, #6
STP #2, @5
CMP <5, $6
STP #2, <7
JMP <0, $1
CMP *4, #6
SEQ {2, {5
JMP {4, @6
DJN  6, #7
JMN @0, {4
SLT }2, @3
SPL #6, <7
DAT @3, <4
CMP  4, *5
JMZ }5, #6
DIV $5, #7
MOD {4,  6
JMP {0, }2
SLT }5, $7
SNE *2, $3
SEQ $4, *7
JMZ {6,  7
MOD <6, *7
ADD }0, $7
DIV }3,  5
SNE $1, #7
JMN *4, $6
JMP @4, }6
LDP $3, #4
MUL *2, }7
DJN @3, $5
CMP $5, {7
LDP @2, $7
SLT <5, }7
JMZ $0, }5
SLT }0, *4
LDP *2, }6
MOV <4, {6
SUB }2,  5
SPL <0, *3
CMP {3, <7
JMN }0, <3
DAT @4, }7
JMN }0,  2
ADD }1, *4
DAT {3,  7
NOP @0, *6
SNE <2, {6
MOD $2, $6
SEQ  4, <6
JMP @6, $7
JMZ *3, *6
LDP #3, @4
SUB }4,  6
SLT @4, <5
SEQ $2, #7
JMZ }3,  7
LDP  2, }6
end ;  execution  ends  here
```

Received March 2017