Quality of Genetic Algorithm in the Cloud

Lee Curry
Department of Computer Science,
University of Texas at El Paso,
El Paso, Texas 79968
Email: lacurry2@miners.utep.edu

Sekou L. Remy
School of Computing,
Clemson University,
Clemson, South Carolina 29634
Email: sremy@clemson.edu

***Abstract*—Genetic Algorithms are a form of artificial intelligence used in adaptive techniques and solution searching. Similar to the notion of natural selection in which the 'most fit' of the population have a greater influence on the following population, genetic algorithms use the most fit of a population to construct the next generation. In terms of use, genetic algorithms are simpler to implement and are one of the more efficient algorithms for complex systems with multiple factors at play. When implementing Genetic algorithms the effects of parameter usage can be unknown when taking the system as a whole into consideration, such a population size, termination criteria, generation length, introduction rate and so on. The focus of this study is to determine if there is a difference in the quality of outcome of a genetic algorithm on different network topologies.**

I. Introduction

Genetic algorithms where first created and developed by John Holland [1]. At the time, his goal was to study the inner workings of adaptation and to find utilization in computers systems. With such a backing, genetic algorithms can be used to quickly and effectively find a solution to any problem with a high complexity. The uses for genetic algorithms have been studied and as a result can be seen in many applications from robotic cloud computing, and searching to optimization [2] [3]. In many situations a set of known solutions would not be feasible to have and store for a later usage with complex systems. In some cases the solution is entirely different from what would be expected, negating the usefulness of a database of working solutions. Genetic algorithms provide an alleviation to this predicament.

As with natural adaption, in which many factors come into play, genetic algorithms have varied parameters that have different effects. Such factors can be said to be the environment in which the system is in, the size of the population, the rate of growth, all of which have a major impact on the outcome. Papers and studies have also been completed with the usage of genetic algorithm parameters as its strong point. [4].

In recent years a new technology has risen to a high standard of use. The cloud, as defined by the National Institute for Science and Technology , " is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources" [5]. Cloud based usage gives a consumer an opportunity to have processing power, storage, use of networks and other computational resources without having to set up, up-keep and support any local machinery [6].

One foreseeable drawback from using cloud based resources is that the underlying network would have a great potential to apply interference across the network. This interference, or noise, has the potential to be harmful to computation or result in unknown outcomes. Even with the knowledge that cloud use can result in efficient use of resources, it is necessary to also know how the cloud can affect the final outcomes of a program, whether the program is time sensitive, complex or neither.

In the reference of this paper a genetic algorithm will be used to determine if the topology of network in which the genetic algorithm is ran will affect the final outcome. The scope of this paper was designed as such that it can be fluidly applied to broader real-time systems.

II. Background

Modeled after the process described the evolutionary theory, genetic algorithm attempt to simulate the growth and change of a set of individuals such that a set of better individuals is produced as an outcome. [7] As with any idea and concept genetic algorithm have many different implementations and structural discrepancies, but at its core all genetic algorithms have the same basic structure. This structure can be broken down into four parts: population sourcing, fitness evaluation, selection, and genetic operations. The following shows and example of a generic genetic algorithm.

```
Initialize population
while: termination criteria not meet
Evaluate population
Elitism
Recombination
Mutation
Introduction
```

A. *Population sourcing*

Population sourcing is the method in which a new individual is selected, or made, to be used in the population. An individual is said to be a vector containing data, this data is used to in the fitness evaluation. Depending on the subject at hand an individual could be constructed using a variety of data[8]. In Holland's original algorithm, a binary vector was to be used. [9] An assortment of 1's and 0's would make up the entirety of an individual's genes. In modern times strings, integers, symbols or any other quantifiable data type could be used.

The process of constructing a new individual could be done entirely random, with a randomization that is suited to meet

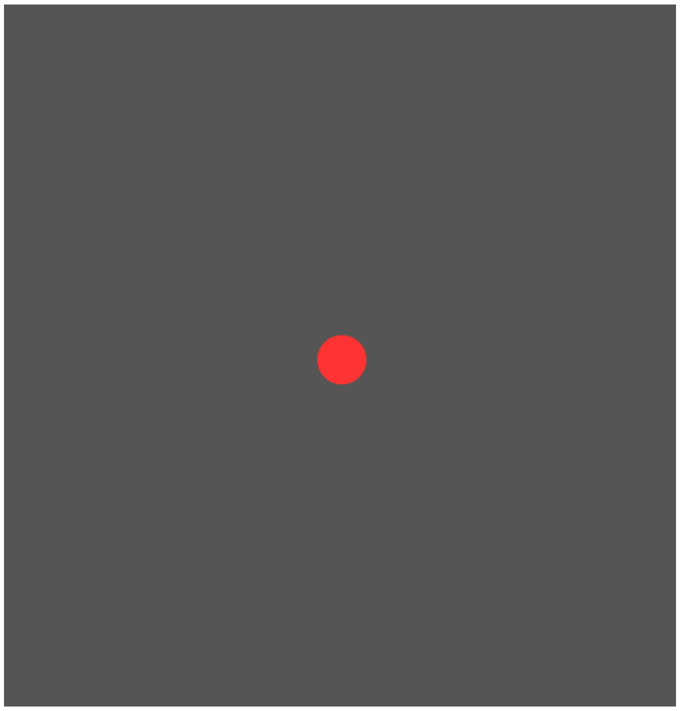

Fig. 1. an example of a ball and plate simulation

a particular standard, or from a file that is constructed for the use of the algorithm.In any case a variety is most used, a generation of individuals that are similar would not be allowed to grow as well as a generation with different individuals.

B. Fitness evaluation

Fitness evaluation is the part of the algorithm that is entirely unique to the situation that calls for it. Albeit one standard for an algorithm such as genetic algorithm is a test which involve an implementations of a genetic algorithms ability to have a ball roll on a plate in a specified pattern , or for a given amount of time. In the case of this study the ball and plate will be simulated, but a physical construction can be used as well. It is important to know that differences in testing result in different outcomes.

In general, an individual is passed through a deterministic test and its fitness score is given. Typically, an individual will have a fitness that can be said to be
$F(x) = y + \alpha$
With α used to denote the amount of noise a system has. This score is used as a factor in the selection for the next population.

C. Population Selection

Population selection is the process in determining what individuals of a population will be chosen to construct the next population. Such as is the case in a natural environment, when one species becomes so dominant to the point of total control, stagnation occurs resulting in the populations swift downfall. To combat this, not all the individuals in the population are ’fit’. In mimicking this property a genetic algorithm has a ratio of elite individuals, fit individuals and unfit individuals.

The individuals that are elite are used to ensure that the evolution does not fall into degradation. With very good individuals ever present, and used for construction of the next population, a declining generation fitness is deterred.

The fit individuals are used to exploit the possibilities of a what good individual can be. It might seem fortuitous to exploit the possibilities of a good individual, but think of a plant with fruit. The fruit my complete the task, but a slight tweak in flavor could be even better for the plant. So to a slight tweak could result in a greater fitness.

The individuals that are unfit are used to expand definition of what it means to be a good individual. Think of two creatures, one with wings and another with webbed limbs. Surly the winged creature is best at flying but if the task was to travel the fastest in any medium, the webbed limbed creature could be better suited for speed in a liquid medium. So such is the use of an unfit percentage in the population.

D. Genetic operations

Genetic operations are methods in which individuals, either in a group or alone, are altered [10]. Two of the most popular methods are recombination and mutation.Though many other exist, these two will be explained in detail .

Mutation is the act of changing one aspect of an individual’s genes. Recombination comes in many flavors, two of which are single point cross over and double point crossover. Single point cross over is the act of taking two individuals and switching every gene after some point in the ’genetic makeup’, i.e. the vector. Double point cross over is the act of taking two individuals and switching the values of the every gene in the ’genetic makeup’ up to some point. Both of these recombination techniques result could result in two new individuals in the next population. This implementation will use the technique of a double point cross over.

Elitism is this reference is seen as the individuals with the highest fitness being saved to the next generation with out any change to the individuals genes. Introduction can be seen as new organisms being added into the environment.

III. Methodology

The genetic algorithm was developed in a Java environment. The program, as a whole, was developed in three parts. A separate program was developed and was used as the ball and plate evaluation. The final program was used to actually simulate a ball and plate. The GA was evolved over 50 generations with each generation having a population size of 40. Earlier run of the GA found that .1 mutation rate resulted in the highest overall final fitness in this implementation. With this knowledge .1 and .15 was used as mutation rates, .1 depicting optimal and .15 depicting suboptimal parameters. The process that was used was repeated using three different topologies: local machines, localized cloud, and cloud.

A. The System

As stated, the system for this experiment, the ball and plate, is used. The ball and plate system is a dynamic set with an equilibrium point that is unstable. An individuals genes are used to control the plate to facilitate the movement of the ball on the plate to move in a square pattern. The genetic algorithm is being used to optimize the parameters for the ball and plate, represented by an individual's genes. Due to the fact that each individual is independent of the other, the evaluation of a generation can be parallelized across multiple threads.

B. The Evaluation

The evaluation was done by calculating the error in the system. If the ball would not complete a remote trajectory that of a square, the error would be quite high, resulting in a low fitness. So to if the error was low, a high fitness would be returned.

C. The Local Run

The genetic algorithm system was first ran on a local machine network, housed inside the computer science department at Clemson University. The System, named Joey, has Ubuntu as an operating system, linked to a network of 27 independent machines. The genetic algorithm, GA for short, was ran on one Joey. The ball and plate evaluation, eval for short, was ran on a different Joey then that of the ball and plate simulation, ball-plate, and the GA.

D. The Localized Cloud

In the localized cloud set, the GA and the eval was ran on different Joeys. There was no difference in topology between these two programs. The only difference in this set was that the ball-plate program was on a different network, connected via cloud. The machine that the ball-plate program was ran on was still in Clemson area, but not physically at the university. This machine was connected to a network similar to that of the Cloud set, but stationed in the Clemson area.

E. The Cloud

The Cloud set, the GA and the eval was ran on different Joeys also. The main difference was that the ball-plate program was now run via cloud. The machine that physically housed the ball-plate program was in Utah. The difference in location was 2727.8 kilometers (1694.97 miles).

IV. Results

The genetic algorithm that was implemented used a double point cross over type of recombination. After each set of runs the data was plotted in two different box plots. One represents the fitness of each generation as a whole, the second is the best top six individuals of the first generation. Part of the implementation was such that all runs had the same initial population, therefore the top six of the first generation is the same individuals in all the top six individual charts.

A. The Local Run

This implementation of a genetic algorithm converged to a final fitness around generation 15. As Fig 2 shows, after generation 15 not much change occurs. The down skew of the plot implies that exploration is still being used, but at the mean number of good individuals does not increase, suggesting that the system is at, or near, its peak performance.

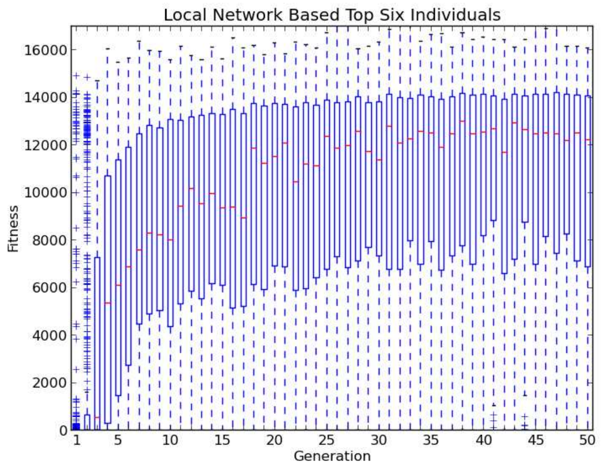

Fig. 2. Ball-plate ran on closed in-house network

B. The Localized Cloud

Comparing the results from this run to those of the local run, a slight shift is seen. This shift is most notable in the curvature of the mean fitnesss. Notice that the curve of Fig 3 is much more concise, while the curve of Fig 2 is more sporadic. Fig 3 shows that the GA converged slightly faster than the local run, but this is marginal. Other than these observations the data of these two run has no statistical difference.

C. The Remote Cloud

Looking at Fig 4 a clear difference can be seen. None of the mean fitnesss are above 2000, while nearly all the fitnesss for each generations after generation 4 on the local and localized run are above about 5000. This though is to be expected. Compared to the local runs there is a large number of outliers, this suggest that there is a large number of relatively higher individual fitnesss; the fact that the space between the first and third quartile are so close means that the number of good individuals was out weighted by the number of bad.

D. The Top Six Individuals

From Fig 5 to Fig 6, there is no statistical difference in the performance of the top six individuals. The skew of the individuals fitness changes, but the location of the mean fitness for that individual does not really move drastically. This observation suggests that running a genetic algorithm on a local machine, as opposed to a localized cloud has no statistical difference. On the other hand looking at Fig 7, a clear difference is seen. The chart for Fig 7 is scaled down so that the largest value is 1000, if not for this little would be seen. With Fig 7 representing the best six individuals from the first generation tested with the Remote Cloud, a clear trend can be witnessed. All the top six individual charts show that the six individual, which is the best individual has a statistical difference no matter where it is ran.

E. Final Outcome

Fig 8 depicts three individuals with the highest fitness of each topology ran on the in-house network 21 times each.

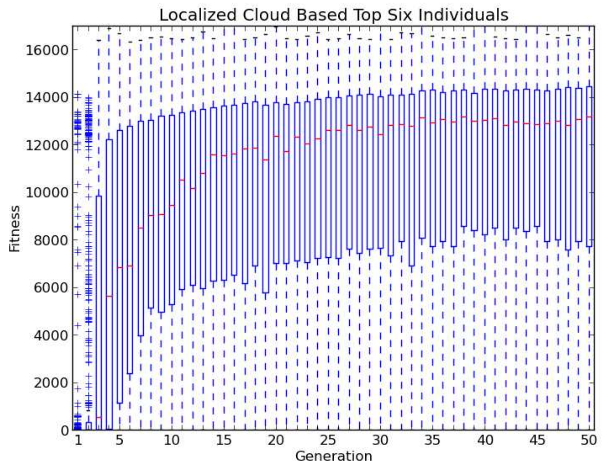

Fig. 3. Ball-plate ran in localized Cloud

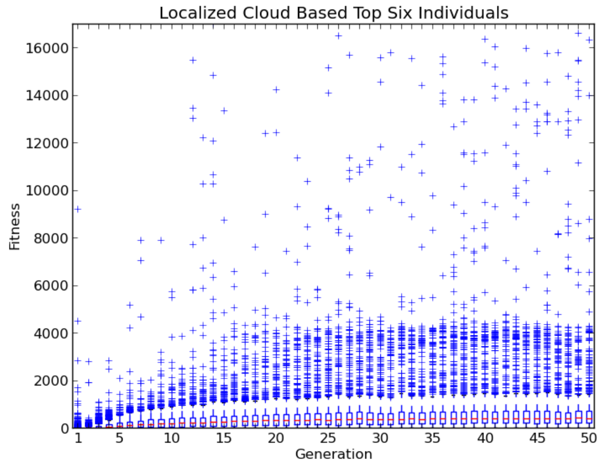

Fig. 4. Ball-plate ran in Remote Cloud

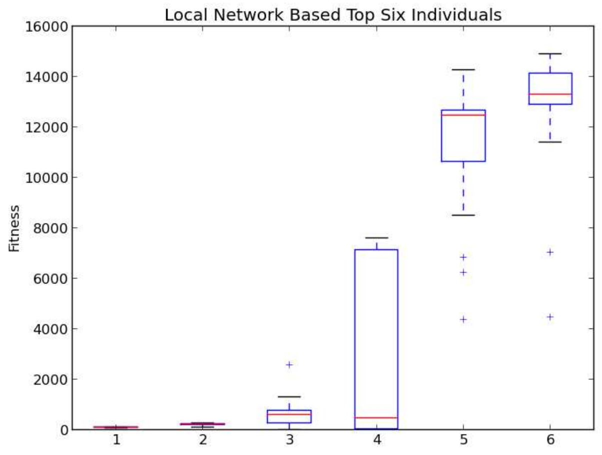

Fig. 5. Top Six Individuals of Generation 1. Ball-plate Ran on closed in-house network

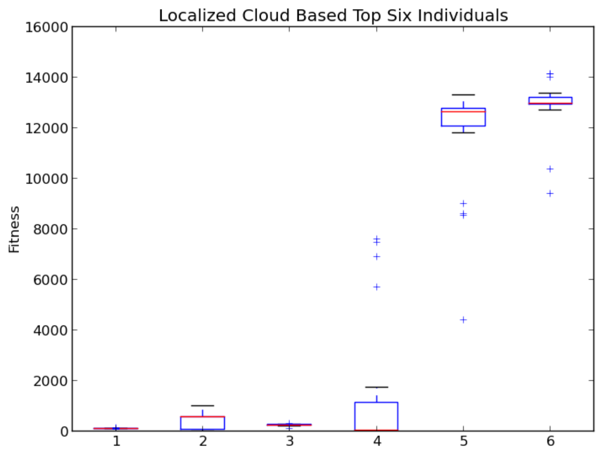

Fig. 6. Top Six Individuals of Generation 1. Ball-plate Ran in Localized Cloud

The chart shows that when the genetic is ran on an in-house network, the results have no statistical difference than when the ball-plate is ran on a localized Cloud. Both topologies are comparable to each other in the sense of near equivalent high fitness's. The main difference comes when comparing either in-house network or localized Cloud to the Remote Cloud usage. There is a clear statistical difference between Remote Cloud as opposed to the other topologies. Usage of a Remote Cloud results in a drastic reduction of fitness. The best performing individuals that came from the Remote Cloud, when ran on the in-house network, performed less then a tenth as well as those that resulting from the other topologies.

V. Conclusion

The final results suggest that using a Localized Cloud to test the individuals will have no statistical impact on the performance of a Genetic Algorithm. Using a Remote Cloud would result in worse performance then is possible. Given the data from the top six individual charts, a good individual will still be good and another topology, but the distance of the evaluation program from the fitness test seems to have influence on the quality. These finding suggest that utilizing a Local Cloud will yield comparable results to using an in-house network when optimizing network controllers and running the evaluation program close to the program that test the individuals will produce higher quality outcomes. The consequences of these comparable results will no doubt shine light on the scalable and elasticity that Cloud usage has to offer, pushing forth growth in this field.

References

[1] M. Mitchell and S. Forrest, "Genetic algorithms and artificial life," *Artificial Life*, vol. 1, no. 3, pp. 267–289, 1994.

[2] K. Kamei, S. Nishio, N. Hagita, and M. Sato, "Cloud networked robotics," *Network, IEEE*, vol. 26, no. 3, pp. 28–34, 2012.

[3] Y.-S. Zhou and L.-Y. Lai, "Optimal design for fuzzy controllers by genetic algorithms," in *Industrial Automation and Control: Emerging Technologies, 1995., International IEEE/IAS Conference on.* IEEE, 1995, pp. 429–435.

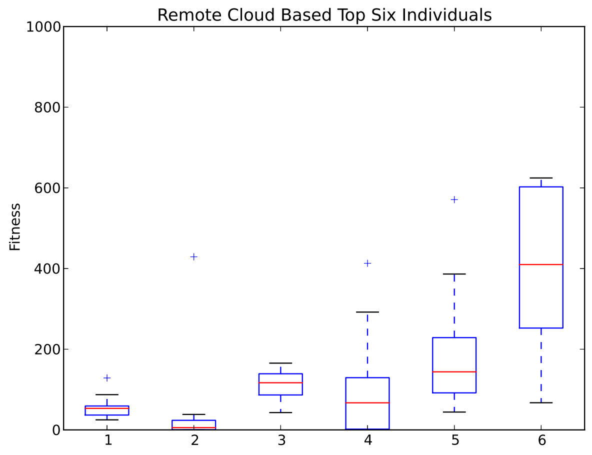

Fig. 7. Top Six Individuals of Generation 1. Ball-plate Ran in Remote Cloud

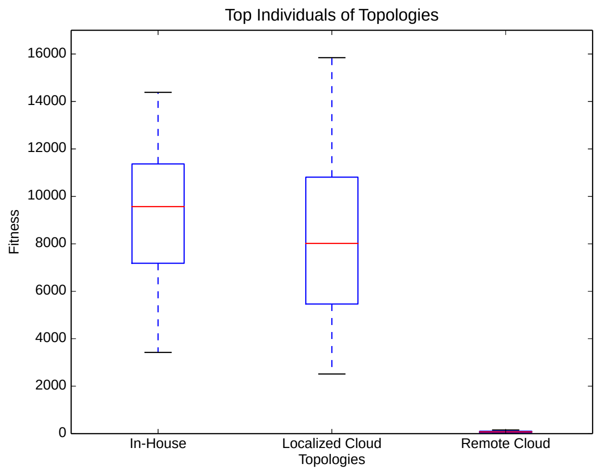

Fig. 8. top three individuals of each topology

[4] R. L. Haupt, “Genetic algorithm applications for phased arrays,” in *ACES*, vol. 21, no. 3, 2006.

[5] P. Mell and T. Grance, “The nist definition of cloud computing,” *National Institute of Standards and Technology*, vol. 53, no. 6, p. 50, 2009.

[6] S. Kaur and A. Verma, “An efficient approach to genetic algorithm for task scheduling in cloud computing environment,” *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 4, no. 10, p. 74, 2012.

[7] D. E. Goldberg and J. H. Holland, “Genetic algorithms and machine learning,” *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[8] C. Wells, C. Lusena, and J. Goldsmith, “Genetic algorithms for approximating solutions to pomdps,” Citeseer, Tech. Rep., 1999.

[9] D. E. Goldberg and J. H. Holland, “Genetic algorithms and machine learning,” *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[10] K. Vekaria and C. Clack, “Selective crossover in genetic algorithms: An empirical study,” in *Parallel Problem Solving from NaturePPSN V*. Springer, 1998, pp. 438–447.