Quantum Evolutionary Programming

Bart Rylander[1] **Terry Soule**[1] **James Foster**[1] **Jim Alves-Foss**[2]

[1]Initiative for Bioinformatics and Evolutionary Studies (IBEST)
Department of Computer Science, University of Idaho, Moscow, Idaho 83844-1014 USA
rylander@up.edu, {soule, foster}@cs.uidaho.edu

[2]Center for Secure and Dependable Software
University of Idaho, Moscow, Idaho 83844-1008 USA
jimaf@cs.uidaho.edu

Abstract

Recent developments in quantum technology have shown that quantum computers can provide dramatic advantages over classical computers for some problems [1] [2]. These quantum algorithms rely upon the inherent parallel qualities of quantum computers to achieve their improvement. In this paper we provide a brief background of quantum computers. We present a simple quantum approach to genetic algorithms and analyze its benefits and drawbacks. We describe the quantum advantage of true randomness. We show that in some cases, such as program induction, there is a measurable difference [3]. These algorithms are significant because to date there are only a handful of quantum algorithms that take advantage of quantum parallelism [4] and none that show an advantage due to true randomness. Finally, we provide ideas for directions of future research.

1 INTRODUCTION

The first major breakthrough in quantum computing came in 1985 with the development of the Quantum Turing Machine (QTM) [5]. Then 1996, two researchers independently proved that a Universal Quantum Simulator was possible [6], [7]. As such, anything computable by a classical computer would be computable by a quantum computer in the same time, if and when such a computer was in fact built. This cannot be said of the converse however. Quantum computers have a feature called quantum parallelism that can not be replicated by classical computers without an exponential slowdown. This unique feature turns out to be the key to most successful quantum algorithms.

Quantum parallelism refers to the process of evaluating a function once on a "superposition" of all possible inputs to produce a superposition of all possible outputs. This means that all possible outputs are computed in the time required to calculate just one output with a classical computer. Unfortunately, all of these outputs cannot be as easily obtained. Once a measurement is taken, the superposition collapses. Consequently, the promise of massive parallelism is offset by the inability to take advantage of it.

This situation changed in 1994 with the development of a fast, hybrid algorithm (part QTM and part TM) for factoring that took advantage of quantum parallelism by using a Fourier transform [1]. With this algorithm and a suitably sized quantum computer it is possible to provide a solution for factoring in polynomial time. This is an important development because the security of most public-key encryption methods relies upon the difficulty of factoring large numbers. Though not proven intractable, factoring had previously seemed secure. Consequently, quantum technology has already made a very tangible impact on some communities.

Previous work that explored the application of quantum parallelism to evolutionary programming has been promising. The first attempt was a "quantum inspired" GA in which many of the operators were modeled after quantum behavior [8]. While this algorithm wasn't actually quantum-based it did imply that a quantum genetic algorithm would outperform a classical one if it could be implemented. The next two attempts showed that a quantum GA maybe possible, but that the

restrictions imposed by the quantum paradigm seemed to negate the parallelism introduced by the evolutionary paradigm [9][10]. We build on these efforts to produce an algorithm that seems to take advantage of both kinds of parallelism. The next section provides a brief introduction to quantum concepts. Section 3 outlines our quantum genetic algorithm (QGA). Section 4 provides an analysis of the advantages of quantum programming. Section 5 details the difficulties that still remain to developing such an algorithm (beyond the obvious fact that a practical quantum computer has yet to be built). Finally, Section 6 gives conclusions and directions for potential research.

2 QUANTUM VS. CLASSICAL

There are two significant differences between a classical computer and a quantum computer. The first is in storing information, classical bits versus quantum *q-bits*. The second is the quantum mechanical feature known as *entanglement*, which allows a measurement on some q-bits to effect the value of other q-bits.

A classical bit is in one of two states, 0 or 1. A quantum q-bit can be in a *superposition* of the 0 and 1 states. This is often written as $\alpha |0\rangle + \beta |1\rangle$ where α and β are the *probability amplitudes* associated with the 0 state and the 1 state. Therefore, the values α^2 and β^2 represent the probability of seeing a 0 (1) respectively when the value of the q-bit is measured. As such, the equation $\alpha^2 + \beta^2 = 1$ is a physical requirement. The interesting part is that until the q-bit is measured it is effectively in *both* states. For example, any calculation using this q-bit produces as an answer a superposition combining the results of the calculation having been applied to a 0 and to a 1. Thus, the calculation for both the 0 and the 1 is performed simultaneously. Unfortunately, when the result is examined (i.e. measured) only one value can be seen. This is the "collapse" of the superposition. The probability of measuring the answer corresponding to an original 0 bit is α^2 and the probability of measuring the answer corresponding to an original 1 bit is β^2.

Superposition enables a quantum register to store exponentially more data than a classical register of the same size. Whereas a classical register with N bits can store one value out of 2^N , a quantum register can be in a superposition of all 2^N values. An operation applied to the classical register produces one result. An operation applied to the quantum register produces a superposition of all possible results. This is what is meant by the term "quantum parallelism."

Again, the difficulty is that a measurement of the quantum result collapses the superposition so that only one result is measured. At this point, it may seem that we have gained little. However, depending upon the function being applied, the superposition of answers may have common features. If these features can be ascertained by taking a measurement and then repeating the algorithm, it may be possible to divine the answer you're searching for probabilistically. Essentially, this is how the famous quantum algorithm for factoring works. First, you produce a superposition and apply the desired functions. Then, take a Fourier transform of the superposition to deduce the commonalties. Finally, repeat these steps to pump up your confidence in the information that was deduced from the transform.

The next key feature to understand is entanglement. Entanglement is a quantum connection between superimposed states. In the previous example we began with a q-bit in a superposition of the 0 and 1 states. We applied a calculation producing an answer that was a superposition of the two possible answers. Measuring the superimposed answer collapses that answer into a single classical result. Entanglement produces a quantum connection between the original superimposed q-bit and the final superimposed answer, so that when the answer is measured, collapsing the superposition into one answer or the other, the original q-bit also collapses into the value (0 or 1) that produces the measured answer. In fact, it collapses to *all* possible values that produce the measured answer. Given this very brief introduction to superposition and entanglement, we can begin to address our GA. (Interested researchers may refer to [4] for a more detailed description of quantum computing.)

3 A QUANTUM GENETIC ALGORITHM

We now present a quantum genetic algorithm (QGA) that exploits the quantum effects of superposition and entanglement. This QGA differs from previous QGA's in several regards. Primarily, our QGA is more similar to classical GA's. This allows the use of any fitness function that can be calculated on a QTM without collapsing a superposition, which is generally a simple requirement to meet. Our QGA differs from a classical GA in that each individual is a quantum individual. For example, consider the fitness landscape in Figure 1.

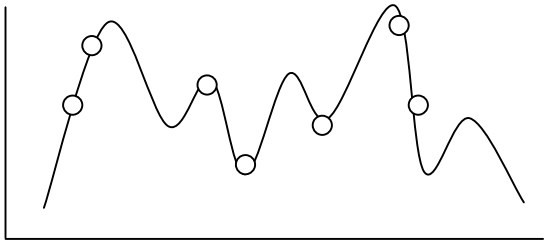

Figure 1: Fitness landscape with individuals

Each point represents a unique position on the fitness landscape. In the case of a fitness function such as multiplication, the fitness of 24 can be produced in a variety of ways. Individuals that have subcomponents such as 6*4, 4*6, 12*2, and 24*1 all have the same fitness despite the fact that they are fundamentally different. Each of these unique individuals will reside somewhere on the landscape. Consequently, when selecting an individual to perform crossover, or mutation, exactly 1 individual is selected. This is true regardless of whether there are other individuals with the same fitness. (See Figure 2)

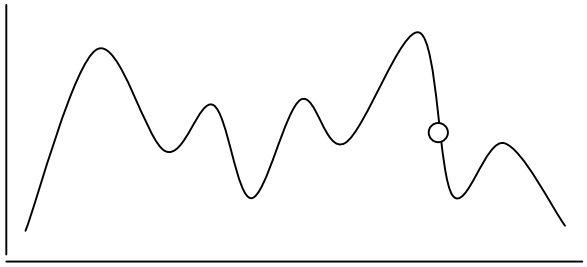

Figure 2: A classical individual is selected

This is not the case with a quantum algorithm. By selecting an individual, *all* individuals with the same fitness are selected. (See figure 3) In effect, this means that a single quantum individual in reality represents multiple classical individuals.

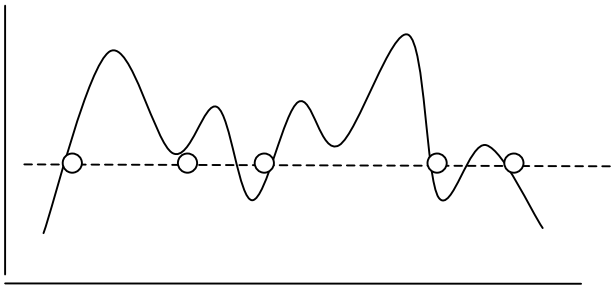

Figure 3: A quantum individual is actually several classical individuals (all those with the same fitness)

In our QGA each *quantum* individual is a superposition of one or more classical individuals. To do this we use several sets of quantum registers. Each quantum individual uses two quantum registers. We refer to these as the *individual register* and the *fitness register*. (See Figure 4) The first register stores the superimposed classical individuals. The second register stores the quantum individual's fitness. At different times during the QGA the fitness register will hold a single fitness value or a quantum superposition of fitness values. A population will be N of these quantum individuals.

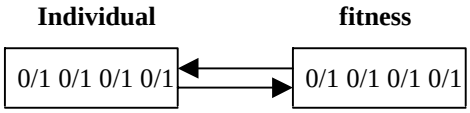

Figure 4: Two 4-qubit entangled quantum registers (representing a single individual)

The key step in our QGA is the fitness measurement of a quantum individual. We begin by calculating the fitness of the quantum individual and storing the result in the individual's fitness register. Because each quantum individual is a superposition of classical individuals, each with a potentially different fitness, the result of this calculation is a superposition of the fitnesses of the classical individuals. This calculation is made in such a way as to produce an entanglement between the register holding the individual and the register holding the fitness(es).

For example, consider a quantum individual consisting of a superposition of six classical individuals with fitnesses: 4, 4, 7, 9, 9 and 11. The fitness calculation will produce a superposition of the values 4, 7, 9, and 11. The values 4 and 9 will have higher probabilities associated with them because classical individuals with those fitnesses occurred more times in the quantum individual.

Next, we measure the fitness. That is, we 'look' in the fitness register. This measurement collapses the register and we measure (or observe) only a single fitness. (Either 4, 7, 9, or 11 in the above example, with 4 and 9 being twice as likely as the other values.) When the fitness superposition collapses to a single value the individual register partially collapses, so that it only holds those individuals with the measured fitness. (In the above example, if we observed a fitness of 4 the individual register would automatically change from holding a superposition of six classical individuals to holding a superposition of two individuals; the two whose fitness is 4.)

This process reduces each quantum individual to a superposition of classical individuals with a common fitness. This allows the selection process to proceed as in

a classical GA, based on a classical fitness function.

Crossover and mutation can then be applied (as previously developed [8]). For the initial population each individual is in a fully mixed state (i.e. contains a superposition of all possible solutions). The fitness calculation described above reduces these individuals to superpositions consisting of all classical individuals with a common fitness. This assures a wide diversity in the initial population.

The complete algorithm is described below:

Generate a population of fully mixed quantum individuals. (Each individual is superposition of all possible classical individuals.)
Calculate the fitness of the individuals.
observe the fitness of each individual. (This collapses the individuals into superpositions of only those classical individuals with the observed fitness.)
Repeat

Selection based on the observed fitnesses.
Crossover and **Mutation**. (Superimposed classical individuals in a single quantum individual will no longer have a common fitness.)
Calculate the fitness of the individuals.
observe the fitness of each individual. (This collapses the individuals into superpositions of only those classical individuals with the observed fitness.)

Until done.

4 ANALYSIS OF THE QGA

Now we can begin to describe what has been gained by this algorithm. Unfortunately comparing convergence times between a GA and a QGA cannot currently be directly performed. This is because there is no current theory to predict convergence time for a QGA. Consequently the convergence times must be compared deductively. Another measure of complexity, the amount of information that is contained in a string, *can* be measured [11]. This type of complexity is more meaningful when evaluating algorithms for program induction, such as Genetic Programming. For this type of complexity, the QGA is superior to the classical GA. Both of these measures will be described in detail below.

4.1 CONVERGENCE TIMES

Currently the answer of whether a QGA converges more quickly than a GA cannot be directly quantified. Only recently has there been a way to characterize the complexity of problems related to *classical* GAs [12]. Since it is currently unknown exactly how this form of probabilistic selection will drive convergence, it would be disingenuous to firmly assert any type of quantitative advantage. Still, it may be possible to gain insights through a discussion of the pros and cons of the QGA in an informal manner.

4.1.1 Increased Diversity

The major advantage for a QGA is the increased diversity of a quantum population. A quantum population can be exponentially larger than a classical population of the same *size* because each quantum individual is a superposition of multiple classical individuals. Thus, a quantum population is effectively much larger than a similar classical population.

This effective size decreases during the fitness operation, when the superposition is reduced to only individuals with the same fitness. However, it is increased during the crossover operation. Consider two quantum individuals consisting of N and M superpositions each. One point crossover between these individuals results in offspring that are the superposition of N*M classical individuals. Thus, in the QGA crossover increases the effective size of the population in addition to increasing it diversity.

There is a further benefit to quantum individuals. Consider the case of two individuals of relatively high fitness. If these are classical individuals, it is possible that these individuals are relatively incompatible; that is that any crossover between them is unlikely to produce a very fit offspring. Thus, after crossover it is likely that the offspring of these individuals will not be selected and their good ‘genes’ will be lost to the GA.

If these are two quantum individuals then they are actually multiple individuals, all of the same high fitness, in a superposition. As such, it is very unlikely that all of these individuals are incompatible and it is almost certain that some highly fit offspring will be produced during crossover. Unfortunately, the likelihood of measuring these individuals may not be very good. Therefore, it is possible that *on average* the quantum algorithm will not have a great advantage. However, at a minimum the good offspring are somewhere in the superposition, which is an improvement over the classical case. This is a clear advantage of the QGA.

It seems likely that the more significant advantage of QGA's will be an increase in the production of good building blocks. In classical GA theory good building blocks are encouraged because statistically they are more likely to produce fit offspring, which will survive and further propagate that building block. However, when a new building block appears in the population it only has one chance to 'prove itself'. Originally a good building

block only exists in a single individual. It is crossed with another individual and to survive it must produce a fit offspring in that one crossover. If the individual containing the good building block happens to be paired with a relatively incompatible mate it is likely that the building block will vanish.

The situation is very different in the QGA. Consider the appearance of a new building block. During crossover the building block is not crossed with *only* one other individual. Instead, it is crossed with a superposition of many individuals. If that building block creates fit offspring with most of the individuals, then by definition, it is a good building block. Furthermore, it is clear that in measuring the superimposed fitnesses, one of the "good" fitnesses is likely to be measured (because there are many of them), thereby preserving that building block. In effect, by using superimposed individuals, the QGA removes much of the randomness of the GA. Thus, the statistical advantage of good building blocks should be much greater in the QGA. This should cause the number of good building blocks to grow much more rapidly. This is clearly a significant benefit.

One can also view the evolutionary process as a dynamic map in which populations tend to converge on fixed points in the population space. From this viewpoint the advantage of QGA is that the large effective size allows the population to sample from more basins of attraction. Thus, it is much more likely that the population will include members in the basins of attraction for the higher fitness solutions.

Interestingly, all of the advantages of our QGA depend on the mapping from individual to fitness. If this mapping is one-to-one then measuring the fitness function collapses each quantum individual to a single solution and the benefits are lost. The greater the degree of "many to oneness" of the mapping from individual to fitness the greater the potential diversity advantage of a QGA.

4.2 QUANTUM GENETIC PROGRAMMING

Another advantage for quantum computers is the ability to generate true random numbers. This becomes important for Genetic Programming (GP) and other methods for automatic program induction. In particular, recent theory work has shown that the output of a GP can be bounded above by the information content of the GP itself [13]. Given this, the advantages of true randomness become readily apparent. By application of Kolmogorov complexity analysis, it has been shown that classical implementations of GPs which use a pseudo random number generator (PRNG) are bounded above by the GP itself whereas with the benefit of a true random number generator, there is no such bound [13].

Briefly, Kolmogorov complexity measures the size of the smallest program that can output a string and then terminate. It is often used when trying to evaluate the information content of static objects (such as strings). (See Definition 1)

> **Definition 1:** $K_S(x) = \min \{|p|: S(p) = n(x)\}$.
> (In this instance, p can be thought of as a program and S as the programming language.) [11]

Essentially, using the tools from Kolmogorov complexity the following Theorem was developed.

> **Theorem 1:** For all strings x, if x is the shortest program that outputs y, that is K(y)=|x|, then $K(x) \geq K(y) + c$.[13]

A graphical depiction maybe helpful. (See Figure 5)

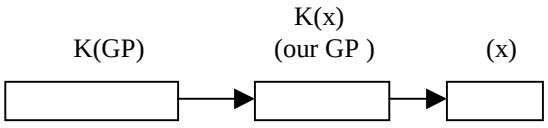

Figure 5: Depiction of the Kolmogorov complexity of a GP and its output

This says that the length of the shortest program to produce our GP must be greater than the shortest program to produce the output of our GP ("x" in this case). However, this theorem does not hold for GPs that have access to a true random source, such as a quantum computer. In particular, for GPs implemented on a quantum computer, Theorem 2 holds.

> **Theorem 2:** For all strings x, y, if x is a shortest length program that outputs y, and x uses a true random source for its generation of y, then:
>
> 1) K(x) is undefined during execution;
> 2) $K(y) \leq |y| + c$; (a well known Kolmogorov result)
> 3) K(y) can be $> K(x)$ - random input.
>
> **Proof:** Let K(x) = n, where x is a GP. Below is x
>
> ```
> returnstring = " "
> for i= 1 to n+1{
>         get a random bit, b
>         returnstring += b
> }
> ```

return returnstring

Since a truly random string is incompressible, K(y) = n+1. Therefore, K(y) > K(x). Since it may be unknown how many times a GP will access a random bit, K(x) is undefined during execution.

It is hard to quantify exactly how great this advantage is. It is clear that the Kolmogorov complexity of the output of a classical GP is bounded by the GP itself. Likewise, it is clear that a GP implemented on a quantum computer has no such bounds. The difference is of course infinite. However, it is hard to fathom how a sequence of truly random numbers could produce such dramatic improvement over an equal length string produced by a PRNG. Trying to verify such an advantage is also difficult since no such practical quantum computer currently exists. Nonetheless despite these difficulties it is a provable advantage.

5 DIFFICULTIES WITH THE QGA

There are some potential difficulties with the QGA presented here, even as a theoretical model. Some fitness functions may require "observing" the superimposed individuals in a quantum mechanical sense. This would destroy the superposition of the individuals and ruin the quantum nature of the algorithm. Clearly it is not possible to consider all fitness functions in this context. However, since mathematical operations can be applied without destroying a superposition, many common fitness functions will be usable.

As noted previously a one-to-one fitness function will also negate the advantages of the QGA. Another, more serious difficulty, is that it is not physically possible to exactly copy a superposition. This creates difficulties in both the crossover and reproduction stages of the algorithm. A possible solution for crossover is to use individuals consisting of a linked list rather than an array. Then crossover only requires moving the pointers between two list elements rather than copying array elements. However, without a physical model for our quantum computer it is unclear whether the notion of linked lists is compatible with maintaining a quantum superposition.

The difficulty for reproduction is more fundamental. However, while it is not possible to make an exact copy of a superposition, it is possible to make an inexact copy. If the copying errors are small enough they can be considered as a "natural" form of mutation. Thus, those researchers who favor using only mutation may have an advantage in the actual *implementation* of a QGA.

6 CONCLUSIONS

We have presented a quantum GA that uses the quantum features, superposition and entanglement. Our simple analysis of the algorithm suggests that it should have three advantages over a normal GA. First, because individuals in the QGA are actually the superposition of multiple individuals it is less likely that good individuals will be lost. Secondly, and more significantly, the effective statistical size of the population appears to be increased. This means that the advantage of good building blocks has been magnified. Presumably this will greatly increase the production and preservation of good building blocks thereby dramatically improving the search process. Finally, in the case of inductive generation of programs, there is a provable improvement over the classical method. Though it is currently hard to evaluate how significant this improvement is, the magnitude of this improvement causes one to wonder what the significance may be in the future.

Unfortunately, the first two advantages can not be presently proven. Therefore, a good direction for future research would include providing a mathematical analysis of the convergence time of the QGA. Once this has been determined, it should be easy to evaluate the relative complexity of QGAs and their classical counterpart. Another potential fruitful direction would be to compare the ease of implementation of crossover methods versus mutation methods. Whereas with classical GAs applying only mutation is in effect a “numerative method” [14] which must contend with the time complexity involved in searching a vast search space, this problem may not exist for a QGA.

Acknowledgments

This paper was supported in part by NSF EPS0080935, NIH F33GM20122-01, and NSA MDA 904-98-C-A894.

References

1. Shor, P. (1994) Algorithms for Quantum Computation: Discrete Logarithms and Factoring, Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124-134

2. Grover, L., (1996) A Fast Quantum Mechanical Algorithm for Database Search, Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, pp. 212-219

3. (2000) On GP Complexity, Proceedings of the Genetic and Evolutionary Computation Conference Workshop Program, pp. 309-311

4. Williams, C., Clearwater, S. ,(1997) Explorations in Quantum Computing. Springer-Verlag New York, Inc.

5. Deutsch, D. (1985) Quantum Theory, the Church-Turing Principle, and the Universal Quantum Computer, proceedings Royal Society London, Vl. A400, pp. 97-117

6. Lloyd, S.,(1996). Universal Quantum Simulators, Science, Vol. 273, pp. 1073-1078

7. Zalka, C., (1998). Efficient Simulation of Quantum Systems by Quantum Computers, Los Alamos National Laboratory, preprint archive, quant-ph/9603026

8. Ge, Y., Watson, L., Collins, E., (1998) Genetic Algorithms for Optimization on a Quantum Computer, Unconventional Models of Computation, Springer-verlag, London

9. Narayan, A., Moore, M., (1998) Quantum Inspired Genetic Algorithms, Technical Report 344, Department of Computer Science, University of Exeter, England

10. (2000) Quantum Genetic Algorithms, Proceedings of the Genetic and Evolutionary Computation Conference, pp. 373

11. Li, M., Vitanyi, P., (1990) Kolmogorov Complexity and its Applications, Handbook of Theoretical Computer Science Volume A. Algortithms and Complexity, pp. 189-254. The MIT Press, Cambridge, Massachusetts

12. (2001) Computational Complexity and Genetic Algorithms, Proceedings of the World Science and Engineering Society's Conference on Soft Computing

13. (2001) Computational Complexity, Genetic Programming, and Implications, Proceedings of the European Genetic Programming Conference