DNA and Quantum Computers

Russell Deaton
Computer Science and Engineering Dept.
The University of Arkansas
Fayetteville, AR 72701
rdeaton@uark.edu
501-575-5590

Abstract

Both DNA and quantum computers have the potential to exceed the power of conventional digital computers, though substantial technical difficulties first must be overcome. Through coherent superposition of states, quantum computers are more powerful than classical Turing machines. DNA computers are evolvable through biotechnology techniques. By combining DNA and quantum computers, both of these characteristics might be captured. DNA computers could be used to self-assemble quantum logic circuits from gates attached to DNA strands. Moreover, quantum computers might be implemented directly using the physical characteristics of the DNA molecule.

1 Introduction

In DNA computing[Garzon and Deaton, 1999], biomolecules and biomolecular reactions are designed to implement computational algorithms. In quantum computing, computation is done at a scale where quantum mechanical effects are important [Bouwmeester et al., 2000]. Both of these relatively new computing paradigms have been mentioned as successors to current solid-state computers [U.S. House of Representatives, 2000]. Both have fundamental advantages over traditional computing, and both have fundamental difficulties to overcome for successful implementation. In this paper, the possibility of quantum and DNA computers working together is explored. The motivation is that each paradigm has unique features that complement the other, and therefore, a system that uses both would have more capability than an implementation of one alone.

The essence of the complementary features of DNA and quantum computing has probably best been expressed by Michael Conrad [Conrad, 1995]. In any computing system, there is a trade-off among programmability, computational efficiency, and evolutionary adaptability. A system is programmable if programs can be communicated to it in an efficient and exact way. Computational efficiency is the fraction of possible interactions in a system that contribute to a computation. Evolutionary adaptability characterizes the system's ability to function in changing and unknown environments. Conventional electronic, quantum, and biomolecular computers each primarily exhibit one of these properties.

Molecular biology techniques and enzymes can be used to evolve biomolecular computers so that they can adapt to changing environments and input. It is the adaptability and robustness of biological systems to external change that has inspired evolutionary programs and artificial neural networks. On the other hand, the complexity of the interactions in a biomolecular computer, and the nature of the biochemistry makes it difficult to program them to behave in a exact way. Moreover, biomolecular computers are not very efficient in their computations. Many interactions are wasted because of errors, or because they do not directly contribute to the desired computational result [Deaton et al., 1998].

Quantum computers excel at computational efficiency because of the entanglement and superposition of an exponential number of states [Bouwmeester et al., 2000]. They can be effectively programmed, though not as easily as conventional computers. Their adaptability, however, is nonexistent because of their extreme sensitivity to the effects of environmental changes. In fact, quantum computer should be isolated from the external environment.

Traditional digital computers fall somewhere in be-

tween biological and quantum computers. They are the most programmable, but are less adaptable than biological and less efficient than quantum.

Therefore, in what follows, first, DNA and quantum computing will be reviewed. Then, several ideas for combining the adaptability of DNA computers and the efficiency of quantum computers will be presented. The presentation is somewhat speculative, of necessity, but the intent is to suggest possibilities, not supply detail blueprints for implementation.

2 DNA Computing and Nanotechnology

Macromolecules of nucleic acids are the prime conveyers of genetic information [Watson et al., 1987]. They are composed of nucleotide building blocks. In DNA, the nucleotides are the purines adenine (A) and guanine (G), and the pyrimidines thymine (T) and cytosine (C). Single-stranded (ss) DNA molecules, or oligonucleotides, are formed when the nucleotides are connected together with phosphodiester bonds. The single strands of DNA form a double-stranded (ds) molecule when the nucleotides hydrogen bond to their Watson-Crick complements, $\overline{A} \equiv T$ and $\overline{G} \equiv C$, and *vice versa*. Oligonucleotides bind in an anti-parallel way with respect to the chemically distinct ends, $5'$ and $3'$, of the DNA molecule. In the DNA helix, the intertwined strands are complementary, and one strand serves as the template for the replication of the other. The base pairing of one oligonucleotide to another is called hybridization. Because of its importance in biology and medicine, DNA is a well characterized molecule, and many standard laboratory techniques exist for its manipulation. Therefore, it is a good choice for nanotechnology and unconventional computing systems.

Adleman [Adleman, 1994] ignited the interest in DNA computing with an actual laboratory demonstration of the computational power inherent in DNA and molecular biology operations. He implemented an algorithm for the solution of a hard, combinatorial problem, the directed Hamiltonian path problem (HPP), which is NP-complete. In HPP, the goal is to find a path through a directed graph that starts and ends at specified vertices, and visits each vertex in the graph once, and only once. There is overwhelming evidence that the problem is impossible to solve by conventional digital computers by means other than a brute-force algorithm. Adleman's insight was to find a way to encode the problem into DNA molecules and turn this algorithm into a feasible search using available biotechnology. The vertices and edges of the graph were encoded in oligonucleotides of DNA so that upon hybridization and ligation, molecules were formed that represented paths through the graph. Because of the massive number of oligonucleotides in the reaction (approximately 3×10^{13} copies of each), the hybridization reactions performed a massively parallel generation of all possible paths in the graph. Therefore, it is the huge number of trials and matches that take place in parallel by the DNA hybridization reactions that represent the brute search power of a DNA computation. After the hybridizations, a molecule representing the Hamiltonian path, if any, would exist in the reaction mixture. Subsequent steps were to extract and identify this molecule. It was shown that this type of DNA search could solve any NP problem in time polynomial in the input size.

Difficulties, however, arose with how the problem instance was represented in the DNA molecules. In the brute force approach, the amount of DNA required to represent the problem grows exponentially with the input size [Lipton, 1995], which lead to some fantastic volumes of DNA to solve problems of interesting size [Hartmanis, 1995]. In addition, for reliable and efficient computation, the hybridizations should take place as planned. Mishybridizations have the potential to produce false positives and negatives through errors that are mistaken as a result, and through wasteful reactions that use up the computational resources available [Deaton et al., 1998]. Though improvements in both these issues have been suggested [Karp et al., 1996, Rose et al., 1999], the economy and reliability of problem representation in DNA sequences continues to be an obstacle to scaling DNA computations to larger sizes.

In response to these difficulties, attempts were made to use the generative power of the random molecular interactions to a computational advantage, rather than regarding them as errors and impediments. Thus, evolutionary approaches were proposed [Deaton et al., 1997] and implemented [Chen et al., 1999], in which mishybridizations were used as sources of genetic diversity, inexact matching generated associative recognition mechanisms, and size was less of an issue because the representation was constantly evolving, and the entire problem space did not need to be represented from the start. Indeed, the fact that DNA can be evolved *in vitro* has produced an entire field of biotechnology. Therefore, the evolvability of DNA is a key property to exploit in biomolecular computers.

Also in response to the problems in the brute

force approach, the self-assembling character of DNA molecules has been exploited for computation and the construction of DNA nanostructures. Based upon Seeman's foundational work [Seeman, 1981] in constructing rigid DNA containers, Seeman, Reif, Winfree and others have used branched DNA molecules [LaBean et al., 2000] with oligonucleotide sticky ends to implement two-dimensional DNA crystals [Winfree et al., 1998]. The lattice was composed of DNA molecular tiles of two types, A and B, which self-assembled through sticky-ended hybridization. These DNA crystals have potential application as nanomechanical devices [Mao et al., 1999], containers for guest molecules, substrates for nanowires, and biocomputing assemblies.

Mirkin *et al.*[Mirkin et al., 1996] introduced the self-assembly of macroscopic structures using DNA hybridization. DNA molecules have been used to direct the assembly of nanoscale semiconductor structures [Coffer et al., 1996], protein assemblies [Niemeyer et al., 1994], and silver wires [Braun et al., 1998]. DNA oligonucleotides attached to colloidal gold particles are used to form nanoparticle assemblies and do diagnostics by color change [Mirkin et al., 1996, Elghanian et al., 1997]. The assembly of the nanostructure was directed by oligonucleotide hybridization.

3 Quantum Computation

Because of quantum mechanics, quantum computers operate in an entirely different way from ordinary computers. For quantum computers, the most important principle is quantum superposition. Suppose that a playing card (ace of diamonds?) can exactly be balanced on edge [Tegmark and Wheeler, 2001]. In the classical world, when the card falls, one of two outcomes are observed, either the card lands face up or face down. In the quantum world, both events happen simultaneously, and therefore, the outcome is a superposition of the classical states. But more than that, the classical states become entangled so that they are correlated with and affect each other. In analogy with conventional "bits" of information, quantum mechanical bits of information are termed qubits, and carry two possible values, $|0>$ and $|1>$. The state of the qubit is given by,

$$|Q>= \alpha|0> +\beta|1>, \tag{1}$$

where $\alpha^2 + \beta^2 = 1$, so that the qubit is in a superposition of both states, and will be "0" with probability α^2 and "1" with probability β^2. Reversible (unitary) operations can be applied to the qubit to implement an algorithm, and a measurement can be done on the qubit in which it assumes a classical value with the probabilities given. To implement a more substantial quantum algorithm, registers of qubits are created which are placed in a coherent superposition of states. Then, operations on the register are effectively operating on all possible states of the register simultaneously. The operations are designed so that the probability of the desired outcome is increased if present. Using this essential paradigm, quantum computers have been shown to be capable of universal computation [Deutsch, 1985]. In addition, it has been shown that quantum computers are capable of doing computations, prime factorization [Shor, 1997] and searching an unsorted list [Grover, 1996], with an efficiency of which conventional computers are not capable. For more details, the reader is referred to [Bouwmeester et al., 2000] and the references therein.

It is interesting to compare the representations of DNA and quantum computers. DNA computers would have individual molecules representing the bit registers. Therefore, if there are n bits in the register, the DNA computer, as well as any computer in the world of classical physics, would need 2^n molecules to represent all possible values of that register in parallel. The power in the DNA computer is that one can fit many molecules into a small volume at once. Nevertheless, the above exponential growth eventually imposes a practical limit on the size of a DNA computer. Quantum computers, however, represent the register as a coherent superposition of its possible states, as opposed to the mixture of states in the DNA computer. Therefore, the quantum computer represents in n qubits what the DNA or classical computer needs 2^n molecules or bits to do, thus, the computational efficiency and power of the quantum computer.

Lloyd [Lloyd, 1993] proposed an implementation of a quantum computer based on transitions between localize electronic states in a heteropolymer. The polymer was assumed to have three constituent monomers, A, B, and C, and was composed of periodic ABC monomer groupings. The interactions among the monomers in the polymer chain would shift the level of the energy levels as a function of its neighbors. Thus, different monomers in different environments (*i.e.* neighbors) would have different excitation frequencies, such as ω_{01}^B, meaning the excitation of the monomer B when its right neighbor, A=0, and its left neighbor, B=1. By applying the appropriate frequency of electromagnetic radiation, data could be input, and transitions induced in the electronic states of the monomers in order to implement a computation. This idea has been implemented using bulk nu-

clear magnetic resonance (NMR) to construct simple quantum computers [Jones and Mosca, 1998]. Several other implementations of quantum computers have been done, including cavity quantum electrodynamics, and trapped ions (see [Bouwmeester et al., 2000]). The main obstacles to practical quantum computing seem to be finding an appropriate material in which to construct large quantum computers, isolating the system from the environment, and practical methods to generate entanglements of a large number of states. Quantum computers also have inherent error potentials from environmental-induced decoherence, which is addressed through specialized error-correcting codes.

4 DNA Self-Assembly of Quantum Computers

Of necessity, quantum logic gates are reversible. A universal quantum gate is the controlled not gate (Figure 1), in which the value of one qubit is influenced by the value of the other, or control qubit. How then to assemble logic circuits from the primitive gates? DNA computers could be used to assemble quantum computers made from other material. The inspiration is the work of Mirkin [Mirkin, 2000] and Winfree. The overall scheme (Figure 2) is that some other nanoparticles implement the quantum gates, and the DNA does the circuit layout. The construction of the circuits would have to be done in such a way that the entanglement of the qubits would be preserved [Benjamin and Johnson, 1995]. DNA is uniquely suited for assembly [Mirkin, 2000]. Arbitrary DNA sequences can be prepared easily, and as in DNA computing, these sequences can be programmed to self-assemble complicated structures. DNA computing techniques could be applied to evolve quantum circuits. Many different types of materials, including quantum dots, can be attached to the DNA [Mitchell et al., 1999]. CdS wires have been assembled directly on DNA templates [Torimoto et al., 1999]. DNA does have a limited temperature range of function, but one could envision that the circuit layout is done at approximately room temperature, and then, the quantum computations done at whatever temperature is required.

5 DNA Quantum Computers

In this section, the possibility of directly using the DNA molecule itself for a quantum computer is explored. There are two schemes for doing this: using NMR of the DNA molecule for computations, or doping the DNA molecule to implement quantum gates.

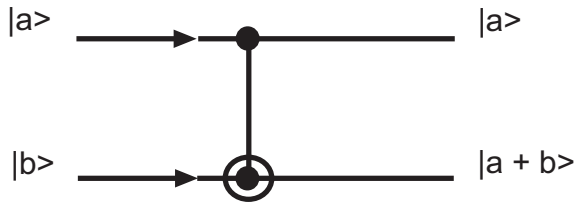

Figure 1: Diagram of CNOT gate.

Table 1: CNOT Truth Table

$\|a>$	$\|b>$	$\|a'>$	$\|a+b>$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Bulk NMR quantum computation[Jones and Mosca, 1998] uses magnetic field induced transitions between the Zeeman levels of atomic nuclei. Spin 1/2 nuclei (^1H, ^{13}C, *etc...*) provide a convenient two-state system for implementing qubits. The frequency response of the nuclei is sensitive to its chemical environment, and therefore, NMR spectra reveal much information about molecular structure. In addition, neighboring spins influence each other, called scalar coupling, which can be used to implement conditional logic, as in CNOT gates. Operations are performed by tailoring RF magnetic field pulses to induce transitions in spins in specific environments. Simple quantum gates have been implemented with bulk NMR using cytosine [Jones and Mosca, 1998] and chloroform molecules [Bouwmeester et al., 2000]. Bulk NMR is usually done near room temperature and in solution. As such, the measurements of spectra are averages over the ensemble of identical spins in the sample. A problem in NMR quantum computations is initializing the ensemble to a uniform state, typically $|0>$ [Bouwmeester et al., 2000]. Several methods for doing this have been proposed [Gershenfeld and Chuang, 1997]. In addition, there is an exponential signal loss with the number of qubits, and decoherence of the coherent superposition of states from random interactions, which at room temperature are greater.

NMR has been used extensively to study DNA molecules [Evans, 1995]. This, at least, suggests the possibility that the ^1H resonance peaks in a DNA

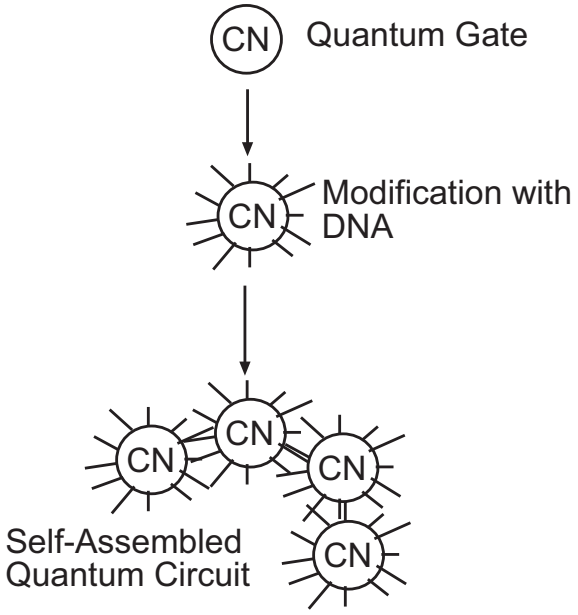

Figure 2: Self-assembly of quantum circuits from nanoparticles which implement CNOT (CN) gates. DNA strands are attached to the nanoparticles, and then, assemble through sequence-specific hybridizations. Sequences can be programmed to assemble specific circuits.

oligonucleotide might be considered for implementing quantum computers in Lloyd's scheme. Theoretically, there is no difficulty with a quantum mechanical description of the state of a DNA molecule, as suggested in [Home and Chattopadhyaya, 1996]. The sequence of the oligonucleotide would be used to determine the local chemical environments and scaler couplings of the particular ^1H's being used as qubits. Then, the evolution and assembly of the quantum computer would proceed through operations applied to the DNA molecule. A problem with using DNA in a bulk NMR quantum computer is that the DNA has too many ^1H peaks. As the length of the molecule increases, the resolution of the peaks degrades, though there have been advances reported extending the range to the hundreds of base pairs [Evans, 1995]. One of these advances is isotopic labeling of the DNA with ^{13}C, which could then be used as the qubit, as suggested at [Kubinec, 1999]. In addition, the number of ^1H resonance centers could be used to establish the initial pure state using the methods of [Gershenfeld and Chuang, 1997].

Another scheme for quantum computation in a DNA molecule is based upon a photo-sensitive repair mechanism in *E. Coli* [Stryer, 1995], which previously was discussed in the context of quantum measurement [Home and Chattopadhyaya, 1996]. When exposed to far UV radiation, DNA is potentially damaged by the formation of covalent bonds between adjacent pyrimidines, forming pyrimidine dimers. A enzyme, DNA photolyase, is produced by *E. Coli*, and upon binding to the damaged region becomes photoactive in the near UV and blue spectral region. When exposed to light in the relevant wavelength region, the dimer is repaired. The formation of the dimer is dependent upon the stacking of the duplex [Cantor and Schimmel, 1980], and thus, somewhat sequence dependent. The frequency response of neither damage nor repair is neighbor dependent [Antony et al., 2000], and therefore, the coupling would have to be accomplished through sequence effects.

Several studies have reported long-range electronic conduction in DNA duplexes[Hall et al., 1996]. It has been postulated that overlap of the π orbitals in the stacked base pair creates a "quantum wire" in which the state of the charge carrier is correlated over long distances. Alternative mechanisms suggest charge hopping among electronic states in intervening bases. The exact mechanism is, as yet, undetermined, but current models[Jortner et al., 1998, Henderson et al., 1999] suggest that there is a mixture of coherent transport and hopping that is sequence dependent[Williams et al., 2000]. The long distance transport is observed by the repair of thymine dimers or guanine oxidization damage. The charge injection is induced by introducing photoactive electron donors and acceptors into the DNA stack.

These characteristics of the DNA molecule suggest that a quantum gate might be implemented directly in the DNA helix by charge injection, conduction, and trapping, modified by sequence. More than likely, the coherent superposition of states could only be done over a relatively short distance in the DNA molecule, as hopping mechanisms seem to dominate over the longer distances. Nevertheless, quantum computers might be implemented using Lloyd's scheme, except that the nearest neighbor couplings would result from sequence-dependent charge transport, and states would be associated with charge injectors and traps. Of course, the qubit in the DNA quantum computer has to exhibit coherent superposition and entanglement. One idea is given in Figure 3, in which a symmetric arrangement is created in the DNA molecule that roughly corresponds to the previously given example of a card balanced on edge. An electron or hole is photo injected into a modified base or metallointercalator in the DNA stack[Ly et al., 1999]. The sequences on either side of the injection point are sym-

metric, and are both terminated by an electron or hole trap, respectively. Given proper engineering of the molecule (*i.e.* choice of sequence), the states of the traps would become correlated. If the traps were a thymine dimer, measurement could be performed by DNA photolyase imaged in an electron microscope, or if guanine damage, by strand scission at the damaged site.

6 Conclusion

There are many obstacles to overcome before any of the suggested combinations of DNA and quantum computing could be successfully implemented. Methods for attaching quantum dots to DNA strands would have to be refined, and circuit design principles developed for DNA self-assembled quantum computers. Effects of the DNA on the quantum circuits would have to be studied and quantified. If DNA itself is used as a quantum computing material, fundamental questions about both NMR and electronic conduction mechanisms would have to be answered. Most importantly, the ability to create a coherent superposition and entanglement of states in the DNA molecule would require both experimental and theoretical validation. Obstacles to this include both the complexity and size of the DNA molecule, as well as the conditions, typically room temperature, at which DNA is manipulated.

In this paper, several ideas for combining DNA and quantum computing have been given. The benefit to such a union might be a computationally efficient computer that can be evolved and adapted using DNA manipulation techniques from biotechnology. Of course, much work, both theoretical and applied, would be involved in approaching such a solution. It may be that the DNA molecule is just too complicated from a quantum mechanical point of view to be a practical platform for quantum computers. Nevertheless, DNA presents a rich range of properties, both from a biological and physical view, that are certainly interesting for implementing nonconventional computers.

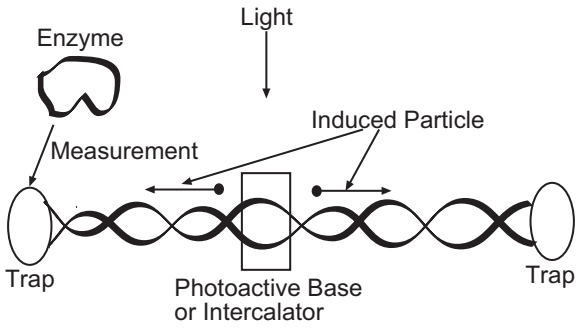

Figure 3: Schematic of DNA implementation of qubit. Light induces a particle (electron or hole) into the overlapping π orbitals of base stacks. Particle traps are located in symmetric positions which induces a coherent superposition of particle states. If trap were thymine dimer, then, state could be measure by protein photolyase.

References

[Adleman, 1994] Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024.

[Antony et al., 2000] Antony, J., Medvedev, D. M., and Stuchebrukhov, A. A. (2000). Theoretical study of electron transfer between the photolyase catalytic cofactor $FADH^-$ and DNA thymine dimer. *J. Am. Chem. Soc.*, 122:1057–1065.

[Benjamin and Johnson, 1995] Benjamin, S. C. and Johnson, N. F. (1995). Entangled electronic states in multiple-dot systems. *Phys. Rev. B*, 51:14733–14736.

[Bouwmeester et al., 2000] Bouwmeester, D., Ekert, A., and Zeilinger, A. (2000). *The Physics of Quantum Information*. Springer-Verlag, Berlin.

[Braun et al., 1998] Braun, E., Eichen, Y., Sivan, U., and Ben-Yoseph, G. (1998). DNA-templated assembly and electrode attachment of a conducting silver wire. *Nature*, 391:775–778.

[Cantor and Schimmel, 1980] Cantor, C. R. and Schimmel, P. R. (1980). *Biophysical Chemistry: Part I The Conformation of Biological Macromolecules*. W. H. Freeman and Company, New York.

[Chen et al., 1999] Chen, J., Antipov, E., Lemieux, B., Cedeno, W., and Wood, D. H. (1999). In vitro selection for a max 1s DNA genetic algorithm. In *Preliminary Proceedings of the Fifth Annual Meeting on DNA Based Computers*, pages 23–37, Providence, RI. DIMACS, American Mathematical Society. DIMACS Workshop, Boston, MA., June 14-16, 1999.

[Coffer et al., 1996] Coffer, J. L., Bigham, S. R., Li, X., Pinizzotto, R. F., Rho, Y. G., Pirtle, R. M., and Pirtle, I. L. (1996). Dictation of the shape of mesoscale semiconductor nanoparticle assemblies by plasmid DNA. *Appl. Phys. Lett.*, 69:3851–3853.

[Conrad, 1995] Conrad, M. (1995). The price of programmability. In Herken, R., editor, *The Universal Turing Machine: A Half-Century Survey*, pages 261–282. Springer-Verlag, Wien.

[Deaton et al., 1998] Deaton, R., Garzon, M., Rose, J. A., Franceschetti, D. R., Murphy, R. C., and Jr., S. E. S. (1998). Reliability and efficiency of a DNA based computation. *Phys. Rev. Lett.*, 80:417–420.

[Deaton et al., 1997] Deaton, R., Murphy, R. C., Rose, J. A., Garzon, M., Franceschetti, D. R., and Stevens Jr., S. E. (1997). A DNA based implementation of an evolutionary search for good encodings for dna computation. In *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, pages 267–272. IEEE. Indianapolis, IN, April 13-16.

[Deutsch, 1985] Deutsch, D. (1985). Quantum theory, the Church-Turing principle the universal quantum computer. *Proc. Roy. Soc. London A*, 400:97–117.

[Elghanian et al., 1997] Elghanian, R., Storhoff, J. J., Mucic, R., Letsinger, R. L., and Mirkin, C. A. (1997). Selective corimetric detection of polynucleotides based on the distance-dependent optical properties of gold nanoparticles. *Science*, pages 1078–1082.

[Evans, 1995] Evans, J. N. S. (1995). *Biological NMR Spectroscopy*. Oxford University Press, Oxfor, UK.

[Garzon and Deaton, 1999] Garzon, M. H. and Deaton, R. J. (1999). Biomolecular computing and programming. *IEEE Transactions on Evolutionary Computation*, 3:236–250.

[Gershenfeld and Chuang, 1997] Gershenfeld, N. A. and Chuang, I. L. (1997). Bulk spin-resonance quantum computation. *Science*, 275:350–356.

[Grover, 1996] Grover, L. (1996). A fast quantum-mechanical algorithm for database search. In *Proc. 28th Annual ACM Symposium on Theory of Computing*, New York. ACM.

[Hall et al., 1996] Hall, D. B., Holmlin, R. E., and Barton, J. K. (1996). Oxidative DNA damage through long-range electron transfer. *Nature*, 382:731–735.

[Hartmanis, 1995] Hartmanis, J. (1995). On the weight of computations. *Bulletin of the European Association for Theoretical Computer Science*, 55:136–138.

[Henderson et al., 1999] Henderson, P. T., Jones, D., Hampikian, G., Kan, Y., and Schuster, G. B. (1999). Long-distance charge transport in duplex DNA: The phonon assisted polaron-like hopping mechanism. *Proc. Natl. Acad. Sci.*, 96:8353–8358.

[Home and Chattopadhyaya, 1996] Home, D. and Chattopadhyaya, R. (1996). DNA molecular cousin of Schrodinger's cat: A curious example of quantum measurement. *Phys. Rev. Lett.*, 76:2836–2839.

[Jones and Mosca, 1998] Jones, J. A. and Mosca, M. (1998). Implementation of a quantum algorithm on a nuclear magnetic resonance quantum computer. *J. Chem. Phys.*, 109:1648–1653.

[Jortner et al., 1998] Jortner, J., Bixon, M., Langenbacher, T., and Michel-Beyerle, M. E. (1998). Charge transfer and transport in DNA. *Proc. Natl. Acad. Sci.*, 95:12759–12765.

[Karp et al., 1996] Karp, R., Kenyon, C., and Waarts, O. (1996). Error-resilient DNA computation. In *Proceedings of the 7th ACM-SIAM symposium on discrete algorithms*, pages 458–467. ACM Press/SIAM.

[Kubinec, 1999] Kubinec, M. (1999). Presentation on quantum
computing. http://waugh.cchem.berkeley.edu/ kubinec/slideshow1/slideshow/sld013.htm.

[LaBean et al., 2000] LaBean, T., Yan, H., Kopatsch, J., Liu, F., Winfree, E., Reif, J., and Seeman, N. (2000). The construction of DNA triple crossover molecules. *Journal of the American Chemical Society*, 122:1848–1860.

[Lipton, 1995] Lipton, R. J. (1995). DNA solution of hard computational problems. *Science*, 268:542–545.

[Lloyd, 1993] Lloyd, S. (1993). A potentially realizable quantum computer. *Science*, 261:1569–1571.

[Ly et al., 1999] Ly, D., Sanii, L., and Schuster, G. B. (1999). Mechansims of charge transport in DNA: Internally-linked anthraquinone conjugates support phonon-assited polaron hopping. *J. Am. Chem. Soc.*, 121:9400–9410.

[Mao et al., 1999] Mao, C., Sun, W., Shen, Z., and Seeman, N. (1999). A DNA nanomechanical device based on the b-z transition. *Nature*, 397:144–146.

[Mirkin et al., 1996] Mirkin, C., Letsinger, R. L., Mucic, R. C., and Storhoff, J. J. (1996). A DNA-based method for rationally assembling nanoparticles into macroscopic materials. *Nature*, 382:607–609.

[Mirkin, 2000] Mirkin, C. A. (2000). Programming the assembly of two- and three-dimensional architectures with DNA and nanoscale inorganic building blocks. *Inorg. Chem.*, 39:2258–2272.

[Mitchell et al., 1999] Mitchell, G. P., Mirkin, C. A., and Letsinger, R. L. (1999). Programmed assembly of DNA functionalized quantum dots. *J. Am. Chem. Soc.*, 121:8122–8123.

[Niemeyer et al., 1994] Niemeyer, C., Sano, T., Smith, C. L., and Cantor, C. R. (1994). Oligonucleotide-directed self-assembly of proteins. *Nucl. Acids Res.*, pages 5530–5539.

[Rose et al., 1999] Rose, J. A., Deaton, R. J., Franceschetti, D. R., Garzon, M., and Stevens, Jr., S. E. (1999). A statistical mechanical treatment of error in the annealing biostep of dna computation. In *Proceedings of the Genetic and Evolutionary Computation Conference, Volume 2*, pages 1829–1834. AAAI, Morgan Kaufmann, San Francisco. Orlando, FL, July 1999.

[Seeman, 1981] Seeman, N. (1981). Nucleic acid junctions: Building blocks for genetic engineering in three dimensions. In Sarma, editor, *Biomolecular Stereodynamics*, pages 269–277, New York. Adenine Press.

[Shor, 1997] Shor, P. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26:1484–1509.

[Stryer, 1995] Stryer, L. (1995). *Biochemistry*. W. H. Freeman and Company, New York, fourth edition.

[Tegmark and Wheeler, 2001] Tegmark, M. and Wheeler, J. A. (2001). 100 years of quantum mysteries. *Scientific American*, 284(2):68–75.

[Torimoto et al., 1999] Torimoto, T., Yamashita, M., Kuwabata, S., Sakata, T., Mori, H., and Yoneyama, H. (1999). Fabrication of CdS nanoparticle chains along the DNA double strands. *J. Phys. Chem. B*, 103:8799–8803.

[U.S. House of Representatives, 2000] U.S. House of Representatives (2000). Beyond silicon-based computing: Quantum and molecular computing. http://www.house.gov/science/basic_charter_091200.htm. Committee on Science.

[Watson et al., 1987] Watson, J. D., Hopkins, N. H., Roberts, J. W., Steitz, J. A., and Weiner, A. M. (1987). *Molecular Biology of the Gene*. The Benjamin/Cummings Publishing Co., Inc, Menlo Park, CA, fourth edition.

[Williams et al., 2000] Williams, T. T., Odom, D. T., and Barton, J. K. (2000). Variations in DNA charge transport with nucleotide composition and sequence. *J. Am. Chem. Soc.*, 122:9048–9049.

[Winfree et al., 1998] Winfree, E., Liu, F., Wenzler, L. A., and Seeman, N. C. (1998). Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394:539–544.

Distributed Virtual Test Tubes

Max H. Garzon*
mgarzon@memphis.edu
Computer Science Division
The University of Memphis
Memphis, TN 38152-3240

Chris Oehmen
coehmen@memphis.edu
Biomedical Engineering
The University of Memphis
Memphis, TN 38152

Abstract

Biomolecular computing (BMC) aims to capture the many advantages that biological molecules have gained in the course of millions of years of evolution for computational purposes, in the same way that evolutionary algorithms aim to capture the key properties of natural evolution. While biomolecules have resolved fundamental problems as a parallel computer system that we are just beginning to decipher, BMC still suffers from our inability to harness these properties to bring biomolecular computations to levels of reliability, efficiency and scalability that are now taken for granted with solid-state based computers. We explore an alternative approach to exploiting these properties by building virtual test tubes in electronics that would capture the best of both worlds. We describe a distributed implementation of a virtual tube, `EdnaCo`*, on a cluster of PCs that aims to capture the massive asynchronous parallelism of BMC. We report several experimental results, such as solutions to the Hamiltonian Path problem (*HPP *) for large families of graphs than has been possible on single processors or has been actually carried out in wet labs. The results show that the paradigm of molecular computing might be implemented much more efficiently (time and costwise) in silico than the corresponding wet experiments. Consequently, we pinpoint a range of practical problem sizes that would make a critical difference in establishing wet biomolecular solutions superior to electronics.*

Corresponding author

1 INTRODUCTION

Biomolecular computing (BMC) is now a fairly known field in computer science. Like genetic algorithms a few decades earlier, it aims to capture the advantages that biological molecules (DNA, RNA and the like) have gained in the course of millions of years of evolution to perform computation out of reach through conventional electronic computers. Several conferences [21, 22, 3, 2, 27] have established the potential of the field to achieve some of this goal, either through new experimental biomolecular protocols using state-of-the-art biotechnology, or through theoretical results, such as universality and complexity, comparing it with the standard, solid-state approach.

It is becoming increasingly clear, however, that the realization of this potential cannot be achieved without addressing the fact that biomolecular protocols in use are too unreliable, inefficient, unscalable, and expensive compared to conventional computing standards. Many currents efforts in the field aim at overcoming these problems by tapping on a number of properties that biomolecules must exercise to accomplish their evolved goal in natural organisms [4, 13]. A good example of such is *in vivo* molecular computing [4], where the protocols are transferred from test tubes to living organisms. While this strategy may produce very interesting results, it presents some shortcomings. First, our understanding of information processing by biomolecules will not increase substantially, just as cloning an organism does not shed any scientific understanding of the complexity of morphogenesis from a computational point of view. Second, the critical issues of reliability, efficiency and scalability remain unanswered for computation *in vitro*.

Just as genetic algorithms and evolutionary computation find their inspiration in the corresponding evolution in natural organisms, an alternative approach to shed light on biomolecular processes is to intro-

duce an analog of biomolecules in electronics and computational algorithms that parallel their biological counterparts. Preliminary attempts have occurred to find good encodings using evolutionary algorithms [6, 14, 28, 11], abstract bases [10] to gauge the reliability of BMC protocols, numerical simulation of reactions [12, 10], and electronic DNA [8, 9] to estimate the performance of a BMC protocol before it actually unfolds in the tube. These analogs have been aimed at addressing specific problems in a pre-processing stage for molecular protocols to be carried out in the wet lab. More recently, we have begun to explore the computational capabilities of this analogy in its own right, somewhat independently of how un/faithfully it may capture the biological processes themselves. Specifically, we have implemented a virtual test tube, called `Edna`, on a single processor [9]. The results reported therein, although preliminary, indicate that `Edna` might be able to effectively solve problems exploiting electronic analogs of DNA molecules.

In this paper we report experiments on a much larger implementation of a virtual test tube in a distributed environment of a cluster of PCs. The distributed implementation, `EdnaCo`, aims to capture the pehonema present in real biomolecules beyond the simulation of their local interactions. These include online genetic algorithms (in the form of subtractive hybridizations) and evolutionary processes simulating in-vitro evolution. We report several experimental results, such as solutions to the Hamiltonian Path problem graphs than has been actually carried out in wet labs, or was possible on `Edna`. We show that electronic DNA is indeed capable of executing *in practice* Adleman's experiment *in silico* for large families containing graphs of arbitrary size. We also give further experimental evidence of the soundness of proposed encoding strategies for biomolecular computation. The results show that the paradigm of molecular computing can be implemented much more efficiently (time and costwise) in silico than the corresponding wet experiments in wet tubes.

The layout of the paper is as follows. In section 2 we given an overview of virtual test tubes. Next we describe in section 3 some details of the architecture of the distributed virtual tube. In section 4 we present some results that allow us to estimate the reliability of biomolecular computations in silico. We argue that this is an upper bound on the reliability of biomolecular computation in-vitro, at least for a relatively large range of problem sizes. Finally, in section 5, we discuss some conclusions of the research presented herein. A point of debate is how far the range of applications possible in virtual test tubes can compete, in practical experiments, with the sizes that can be tackled in wet labs.

2 VIRTUAL TEST TUBES

Perhaps the most fundamental issue to address is to identify the properties that confer on biomolecules their purported superiority as a computational medium. This question may be difficult to answer. For example, it is conceivable that the duality exhibited by many fundamental particles at several levels (muons and gluons at the quantum level, for example) may afford interesting computational media (a form of quantum molecular computing, for example). In this paper, however, we will assume a strong form of the Church-Turing Thesis, i.e., we will bar the existence of such elemental computational steps achievable only through nucleic acids *in vivo*. That is, we will assume that the competitive advantage of biomolecules resides exclusively in their massive parallelism that could be, in principle, achieved on conventional media (solid-state electronics).

On this assumption, what are then the fundamental advantages of biomolecules? Some natural candidates emerge when comparing the behavior of a biomolecular ensemble in a test tube to a parallel computer. The type of computation is *asynchronous, massively parallel* and determined by both local and global properties of the processor molecules. Biomolecules seem to have solved very efficiently key problems such as communication, load balancing, and decentralized control. This seems to be achieved through highly localized base-to-base interactions, template matching, and enzymatic reactions, mediated by randomized motion in both transport and reactions. These characteristics are reminiscent of cellular automata, although their sites are rather asynchronous, randomized, and not localized in space but in mobile molecules. The unfolding of molecular computations is also reminiscent of genetic algorithms driven by a natural fitness function defined by the Gibbs free-energy of hybridization. It is therefore not surprising that cellular automata and molecular computing face a common fundamental problem, namely, a *programming methodology* that would unleash their power to solve problems now considered too difficult for conventional machines. Despite half a century of research, genetic algorithms seem to be the only useful methodology in sight to address this problem for cellular automata [24]. This line of reasoning suggests that an appropriate analog may thus capture some of these properties in an electronic computational medium. We will restrict our attention in this paper to conventional solid-state devices,

simply because the basic problems of reliability and implementation have matured to satisfactory levels in conventional computers.

Edna's architecture has been described in detail in [9]. In short, Ednais a piece of software that simulates the reactions that might actually happen in a test tube, full with all the random brownian motion and chains of complex molecular interactions. Ednathus acts as a virtual test tube where electronic DNA molecules undergo local interactions governed by local rules. By varying the local rules, we show what may be seen as significant evidence that electronics can also exhibit a number of interesting advantages as a biomolecular computer, including ready programmability, robustness, and a high degree of reliability. The simulation is based on purely local molecular interactions between cells in a cellular automaton-like space. The space is an array of cells. The cells represent quanta of 2D or 3D space that may be empty or occupied by nucleotides, solution molecules, and/or other reactants. Each cell is also characterized by associated parameters that render the tube conditions in a realistic way, such as temperature, salinity, nucleotides, covalent bonds, etc. The cellular space can be set up to boundary conditions that reflect tube walls. The transitions of the automaton implement the massive random motion of single stranded molecules compounded over space and time, their multiple attempts and eventual success at hybridizing into double stranded molecules according to various hybridization rules, under various temperatures and reaction conditions and for a variety of input encodings. Currently, two models of hybridization are implemented, one based on the well-known nearest-neighbor model ([23], Gibbs free-energy [26]), and another based on a more computational criterion, the h-distance introduced in [8]. Ednahas an ergonomic interface to choose among these local rules for hybridization, to place and remove strands in the tube, and in general, to set up the various reactions conditions in the desired combination.

Once the initial conditions of the experiment have been set up, the virtual tube comes to equilibrium under the local rule of interaction specified, showing a possible outcome of the experiment. That the simulation bears a significant resemblance to the analogous experiment if it were to be implemented in the wet tube running the the underlying physical chemistry, has been argued theoretically in [9] and evidenced by the experimental results therein. The results reported below provide further experimental evidence since the distributed implementation EdnaCocaptures more faithfully reaction conditions in a wet tube environment.

3 EDNACO'S ARCHITECTURE

Ednaco is a distributed environment for simulating Brownian motion of single-stranded DNA fragments for the purpose of simulating bomolecular computations. The computational framework of Ednaco is a set of data structures which are distributed over several processing nodes which are joined, transparently to the user, in order to produce a single tube simulator in each run. The underlying structure on each node is a tube fragment consisting of a "node table", a two or three dimensional array containing information about the location of each strand and reaction conditions. The node table also tracks empty nodes to provide space for movement of the strands. Because of the tube's absolute coordinate system, the node table is itself meaningless without strands, each of which is a data structure representing a biomolecule. The strand structure also contains information about the location of the nucleotides, which again is meaningful only in the context of the node table. Strands are tracked by a report structure. It is a sorted list of strands by nucleotide length. It checks forward and reverse strand orientations, and keeps track of deletions (when a strand leaves the local tube fragment) and additions (incoming strands, vertex additions at the outset of the simulation, and hybridizations). The entire simulation is distributed over a cluster of (currently up to 8) processors in such a way that each local processor holds an entire tube fragment. This tube fragment is a linked list of strand head nucleotides, and is iterated through for the purpose of moving the strands randomly. If a strand tries to leave the left or right boundary of the fragment, it is sent via message passing to the "adjacent" process, and randomly placed in that tube fragment. The tube fragments are conceptually strung together to produce a coherent tube structure. This is crucial to the overall functioning of EdnaCosince partial solution strands must be able to potentially hybridize with *all* vertex encodings, many which may reside outside the local tube fragment.

During a simulation, when a strand head randomly "walks" into an occupied node containing a nucleotide in a different strand, hybridization is considered. There are two primary criteria for hybridization: 1) stacking energy and 2) h-distance (defined below). Stacking energy takes into account all the currently unhybridized nucleotides on the ends of the strands and calculates the new hybrid with the lowest possible stacking energy. If this hybrid has an energy above the threshold (as determined by the user), hybridization does not take place, and the strands are simply allowed to continue on their random motion. If the

stacking energy of the best hybrid is below the threshold, hybridization takes place, a new hybrid is randomly placed in the tube fragment, and also copies of both original strands still remain in the tube. In the second mode, the h-distance metric is likewise used to determine the best possible alignment of the two meeting strands in a hybrid. If the h-distance of the best hybrid is above the threshold (as determined by the user), hybridization does not take place, as in the stacking energy case. If the h-distance of the best hybrid is below the threshold, a hybrid is created and randomly placed in the tube fragment.

The computer interface of `EdnaCo`is a primitive two dimensional boolean (text) array. The columns indicate strand length. Each row is a time slice consisting of a virtual gel that summarizes the tube content by strand length and multiplicity. The tube content of `EdnaCo`can actually be interfaced with `Edna`for a more realistic visualizatiuon of the entire distributed test tube.

4 EXPERIMENTAL RESULTS

Now we present a summary of a number of experimental runs obtained by implementing several biological protocols in `EdnaCo`. `EdnaCo`has been protoyped on the Hamiltonian Path Problem [1]. The molecules represent graph vertices and directed edges, and the hybridization logic carries the brunt of the computational process to produce longer molecules representing paths in the graph. Once the chemistry reaches equilibrium, it is a matter of searching the products of the reaction to determine whether a molecule exists representing the witness Hamiltonian path. Adaptions for other problems are readily made and experiments for other problems (such as MAX-CLIQUE) are being conducted that will be reported elsewhere.

4.1 SCALING UP ADLEMAN'S EXPERIMENT

The first problem was to reproduce and scale up the original experiment performed by Len Adleman in a real test tube with real DNA molecules [1]. On `EdnaCo`, we were able to systematically reproduce Adleman's result with electronic versions of his original encodings and several other encodings that were deemed good acoording to two proposed measures, the computational incoherence and the h-distance, described below. We succeeded with very sparse graphs (such as paths and cyclic graphs). `EdnaCo`established (non)Hamiltonicity without a problem with a 100% reliability for cycle up to 20 vertices. The results appeared perfectly scalable to any cycle size, given sufficient hardware.

Next, we scaled up the number of edges in the problem instance in order to run in silico an experiment that no one in the literature has reported, in order to test, in the small range allowed by the small cluster (8 processors), the potentially enormous scalability of Adleman's approach that makes DNA-based computing so fascinating. We were successful to scale the results systematically on `EdnaCo`up to about 10 vertices on sparse graphs (up to about 15 edges). For graphs with many more edges, numerous partial paths were produced but `EdnaCo`failed to show the formation of a Hamiltonian path, which was expected. A sample of the results is shown in Table. 1 for graphs with 5 vertices. The "mode" refers to the type of hybridization rule (E = Thermodynamic - threshold in Gibbs free energy; H = h-distance mode - threshold in h-distance). "Path" refers to the witness path actually formed. Numbers in ()'s indicate an edge to or from the vertex number indicated, but no vertex encoding on the hybrid. "C" refers to a cyclic graph, "K" refers to a complete graph and "G" refers to the graph with 5 vertices $0, 1, 2, 3, 4$ and directed edges $0 \to 1, 0 \to 3, 1 \to 2, 1 \to 4, 2 \to 3, 3 \to 2$ and $3 \to 4$. We are currently refining the implementation of a method of subtractive hybridization so that paths are more systematically groomed to prevent false negative partial paths from swamping the tube, by one of several proposed strategies [21]. We believe these techniques will be sufficient to close the gap. Finally, despite attempts to solve instances of sparse graphs of size 100 vertices, we are unable to report success on the current cluster (where the maximum run can only last 36 hours), but the partial paths being obtained are encouraging.

Table 1: `EdnaCo`**solution to** HPP **on 5 vertices.**

Encoding	Path found	Run Time
28g1/E/C	$0 > 1 > 2 > 3 > 4$	55 min.
28g2/E/C	$0 > 1 > 2 > 3 > 4$	11 min.
28g1/H/C	$0 > 1 > 2 > 3 > 4$	50 min.
28g2/H/C	$0 > 1 > 2 > 3 > 4$	11 min.
28g1/E/K	cyclic	29 min.
28g2/E/K	$(4) > 1 > 2 > 4 > 3 > (4)$	22 min.
28g1/H/K	$(3) > 0 > 4 > 1 > 2 > (4)$	82 min.
28g2/H/K	$(4) > 3 > 1 > 4 > 0 > (3)$	26 min.
28g1/E/G	$0 > 1 > 2 > 3 > 4$	< 4 hrs.
28g2/E/G	$0 > 1 > 2 > 3 > 4$	41 min.
28g1/H/G	$0 > 1 > 2 > 3 > 4$	< 4 hrs.
28g2/H/G	cyclic	42 min.

Based on this data, we estimate that the probability

of success of an HPP run is about $15/16 \equiv 94\%$ for a graph in this range. (Computations with more runs show similar figures) We estimate that, on larger state-of-the-art clusters running the same implementation of `EdnaCo`, we will realistically solve random instances of sparse graphs with over 200 vertices with a comparable degree of reliability and within reasonable times. Note that the sparse region of HPP is the range of interest since sufficiently dense graphs are guaranteed to be Hamiltonian by any of well known sufficient conditions (for example, Dirac's test on the number of edges).

4.2 EVALUATION OF CI ENCODINGS

The experiments also permit an analysis of the encoding used in DNA computations. A systematic comparison of encoding quality similar to that reported in [9] for graphs of 5 vertices was done with larger families of graphs.

The first measure for encoding goodness tested was the computational incoherence, ξ, based on statistical mechanics (see [19, 20] for a precise definition). The measure is based on the average probability, at equilibrium, that a randomly observed hybridization within the annealed mixture is in an error configuration with respect to the computation. Using a standard genetic algorithm with $-\log_{10}\xi$ as the applied measure of fitness, sets of DNA words of different lengths were evolved for a Hamiltonian Path problem. Use of this fitness produced encodings with small probabilities of error. Alternatively, the fitness was maximized to produce encodings with large probabilities of errors. In addition, encodings with intermediate probabilities of error were also produced. The values of ξ for the different combinations produced is shown in Table 2. The specific encodings are not shown due to space limitations.

Table 2: Encoding Quality according to CI fitness.

Quality	Length	ξ
Good ξ	12	2.34×10^{-7}
Medium ξ	12	8.56×10^{-4}
Bad ξ	12	1.0
Good ξ	20	1.15×10^{-10}
Medium ξ	20	1.24×10^{-6}
Bad ξ	20	1.0
Good ξ	28	3.94×10^{-16}
Medium ξ	28	6.03×10^{-6}
Bad ξ	28	1.0

These encodings were used for both the virtual tube simulations of ξ as a measure of encoding goodness. The results of the runs are shown in Table 3.

Coding	H-rule	Thr.	Space	HP	Hrs.
CI12g	E	4	40*40	Y	4
CI12g	H	6	40*40	N	4
CI12m	E	4	40*40	N	4
CI12m	H	6	40*40	N	4
CI12b	E	4	40*40	N	4
CI12b	H	6	40*40	N	4
CI20g	E	4	50*50	Y	12
CI20g	H	10	50*50	N	12
CI20m	E	4	50*50	N	12
CI20m	H	10	50*50	N	12
CI20b	E	4	50*50	N	12
CI20b	H	10	50*50	N	12
CI28g	E	4	60*60	Y	36
CI28g	H	14	60*60	N	36
CI28m	E	4	60*60	N	36
CI28m	H	14	60*60	N	36
CI28b	E	4	60*60	N	36
CI28b	H	14	60*60	N	36

Table 3: `EdnaCo`runs on CI-encodings.

In the `EdnaCo`runs on the CI encodings, molecules generally were formed faster with the free-energy hybridization condition than the H-metric condition. With the energy condition, the good encodings for all lengths produced Hamiltonian paths. This occurred probably fairly quickly from the start of the run, which lasted 4 hours (in the current setup, we have no way to time more precisely). Very few error hybridizations were observed. The medium quality encodings primarily produced molecules that were in the proper hybridization frame, although not long enough. More mishybridizations, however, were observed with the medium than with the good quality encodings, and it is unlikely that longer runs might have produced a Hamiltonian path. The bad quality encodings produced many mishybridizations, and much fewer hybridizations in the proper frame. After a 4 hour simulation, it was evident from the molecules formed that no Hamiltonian path was possible. With the H-metric hybridization condition, the CI encodings did much worse. No Hamiltonian paths were formed for any of the encodings. Most of the oligonucleotides hybridized in improper frames. In addition, for hybridization thresholds less than half the encoding length, no hybridizations formed. Once a threshold equal to half the length was reached, hybridization occurred quickly, but in error modes.

4.3 EVALUATION OF h-METRIC ENCODINGS

A more computational measure of hybridization likelihood has been introduced in [7] that extends Hamming

distance (defined as the difference between the number of WC matching pairs from the common length of lined up strands). Hamming distance is not good enough for DNA-based computations in solution since it ignores likely frame-shifts in the tube. The h-measure [7] between two oligos x and y is defined as the minimum of all Hamming distances obtained by successively shifting and lining up the WC-complement of y against x. A small measure indicates that the two oligos are likely to stick to each other one way or another; a large measure indicates that *under whatever physico-chemical conditions* y finds itself in the proximity of x, they are far from containing many WC complementary pairs (let alone segments), and are therefore less likely to hybridize, i.e., they are more likely to avoid an erroneous unwanted hybridization. The maximum length of these segments is controlled by a parameter τ, that is a fairly coarse expression of the reaction conditions.

We ran a similar test of encodings evolved using the h-metric as fitness function. The results of the runs are shown in Table 4. The encodings were evolved used `EdnaCo`'s on-line genetic facility as well.

Table 4: `EdnaCo`Runs on H-encodings.

Coding	H-rule	Thr.	Space	HP	Hrs.
H12g	E	4	40*40	Y	4
H12g	H	4	40*40	Y	4
H12m	E	4	40*40	N	4
H12m	H	4	40*40	N	4
H12b	E	1	40*40	N	4
H12b	H	1	40*40	N	4
H20g	E	4	50*50	Y	12
H20g	H	4	50*50	Y	12
H20m	E	4	50*50	N	12
H20m	H	4	50*50	N	12
H20b	E	4	50*50	N	12
H20b	H	4	50*50	N	12
H28g	E	4	60*60	N	36
H28g	H	4	60*60	N	36
H28m	E	4	60*60	N	36
H28m	H	4	60*60	N	36
H28b	E	4	60*60	N	36
H28b	H	4	60*60	N	36

The results were similar to those obtained for the CI-based encodings. In case no Hamiltonian path was obtained, the time indicates how long they were run before giving up. As expected again, the good encoding produced the desired path fairly quickly regardles of the hrybridization rule used, while bad encoding produced no results.

5 CONCLUSIONS

The results reported here show encouraging evidence that electronic DNA is capable of solving *in practice* Adleman's experiment *in silico* for fairly large problem sizes. It also confirms what is now accepted, that not all encodings (word designs) are equal when it comes to experimental performance in terms of reliability of biomolecular computations. Our results give experimental evidence of the quality of encoding strategies based on theoretical analyses. We note also that these results show a fairly high correlation between the two criteria for encoding quality. A more careful quantitative comparison is underway and will be reported elsewhere.

On a large scale, although it is clear that the implementation in conventional electronics places a limitation on the range of solutions (as does implementation in real life test tubes), several advantages to the simulation approach emerge from from these experiments. First, it is clear that the savings in cost and perhaps even time, at least in the range of feasibility of electronic DNA, are enormous compared to the cost and time it would take ordinary lab protocols, which, to our knowledge, no one has attempted after Adleman's original experiment. Second, as mentioned before, electronic DNA achieves these solutions circumventing current problems of reliability and control. The physics and chemistry is virtual and therefore, somewhat more programmable. Moreover, as mentioned above, it is possible that the problems of scalability posed by conventional electronics itself could now be solved by finding different physical implementations of electronic DNA (at the quantum level, for example) once there is enough evidence of the type shown here with `EdnaCo`, that the paradigm works for small sizes of DNA.

In a different direction, the work presented here further confirms what was already advanced in [9], namely, that massive parallelism is not the only true source of power in biomolecular computations. With hindsight, this is not surprising. Randomness and noise appear to be productive forces in biological evolution and development. Computational systems like genetic algorithms and stochastic Hopfield networks have imitated the randomness inherent in natural systems to provide more efficient mechanisms for searching, adaptability, and robustness. In molecular computers, randomness in the reactions is inherent. Also inherent is the bounded volume and reaction conditions in which the reactions must take place. Like for hidden layers in neural networks and limited individual life spans in genetic algorithms, *bounded resources* force the devices

to improve efficiency. This appears to be the case in biomolecular computation as well.

Last but not least of all, since the electronic DNA paradigm is known to be computation universal in principle, we obtain alternative parallel algorithms that exploit the inherent parallelism of natural processes without facing the problems of synchronization and load-balancing that afflict current electronic parallel machines. In particular, this type of solution could be scaled to a massive environment (in principle as large as the internet) with moderate effort. Finally, it is easy and natural to couple `EdnaCo`with evolutionary computation to implement in-vitro evolution anew *in silico* in a much more controlled and reliably environment and at scales much larger than currently possible.

Acknowledgments

The authors would like to thank JICS, The Joint Institute for Computational Science of the University of Tennessee-Knoxville, for making their PC cluster environment available to implement `EdnaCo`. Thanks also go to John Rose at The University of Tokyo for providing some of the encodings that were used in the simulations reported here, and to Russell Deaton for stimulating conversations.

References

[1] L. M. Adleman, Science, **266**, 1021 (1994).

[2] *Proc. of The Genetic and Evolutionary Computation Conference GECCO-99*, W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Hanovar, M. Jakiela,. R.E. Smith, (eds.), 1999, Morgan Kaufmann.

[3] A. Condon, G. Rozenberg (eds.), Preliminary Proc. of the 6th International Meeting on DNA-based Computers. Leiden University, The Netherlands, 2000.

[4] T.L. Eng, "On Solving 3CNF-satisfiability with an in-vivo algorithm". In [21], 135-141.

[5] R. Deaton, M. Garzon, R. E. Murphy, J. A. Rose, D. R. Franceschetti, S. E. Stevens, Jr. The Reliability and Efficiency of a DNA Computation. Physical Review Letters **80**, 417 (1998).

[6] R. Deaton, R. E. Murphy, J. A. Rose, Max Garzon, D. R. Franceschetti, S.E. Stevens, Jr. A DNA based Implementation of an Evolutionary Search for Good Encodings for DNA Computation. Proc. IEEE Conference on Evolutionary Computation ICEC-97, 267-271.

[7] M. Garzon, P. Neathery, R. Deaton, R.C. Murphy, D.R. Franceschetti, S.E. Stevens, Jr.. A New Metric for DNA Computing. In [15], 472-478.

[8] M, Garzon, R. Deaton, J.A.Rose, D.R. Franceschetti, Soft Molecular Computing, Proc. of the 4th workshop, Princeton University, 1998. In [22], 89-98.

[9] M, Garzon, R. Deaton, D. Renault, Virtual Test Tubes: a New Methodology for Computing. Proc. 7th Int. Symposium on String Processing and Information Retrieval. A Coruña, Spain. IEEE Computer Society Press, 2000, pp. 116-121.

[10] A. Nishikawa and M. Hajiya. Towards a System for Simulating DNA Computing with Whiplash PCR, Proc. of the Congress on Evolutionary Computation CEC-99.

[11] A. J. Hartemink, D. K. Gifford, Thermodynamic Simulation of Deoxyoligonucleotide Hybridization of DNA Computation. In [21], 25-38.

[12] A. J. Hartemink, T. Mikkelsen, D. K. Gifford, Simulating Biological reactions: A Modular Approach. In [22], 109-120.

[13] S. Ji, "The Cell as the smallest DNA-based Molecular Computer". In [21], 123-133.

[14] J. Khodor, D.K. Gifford, A. Hartemink (1998), Design and Implementation of Computational Systems Based on Programmed mutagenesis. In [21], 101-107; 287-297.

[15] J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba, R.L. Riolo, eds. (1997). *Proc. 2nd Annual Genetic Programming Conference*, Morgan Kaufmann.

[16] J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba, R.L. Riolo, eds. (1998). *Proc. 3rd Annual Genetic Programming Conference*, Morgan Kaufmann.

[17] L.F. Landweber, E.B. Baum (eds.), *DNA Based Computers II*, Proc. of the 2nd workshop, Princeton University, 1996. DIMACS series of the American Mathematical Society, vol. 44 (1999), 247-258, Providence RI.

[18] R. Lipton, E. Baum (eds.) *DNA Based Computers.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. **27**. American Mathematical Society, Providence RI.

[19] J. A. Rose, R. Deaton, D. R. Franceschetti, M. Garzon, S. E. Stevens, Jr., A Statistical Mechanical Treatment of Error in the Annealing Biostep of DNA Computation. Special program in DNA and Molecular Computing at the Genetic and Evolutionary Computation Conference (GECCO-99), Orlando FL, July 13-17, 1999. Morgan-Kaufmann, 1829-1834.

[20] J. A. Rose, R. Deaton, D. R. Franceschetti, M. Garzon, and S. E. S. Jr., Hybridization error for DNA mixtures of N species. to *Physical Review Letters*, submitted.

[21] L. Kari, H. Rubin. D. Wood (eds.), 4th DIMACS workshop on DNA Computers. Special Issue of the *J. of Biological and Information Processing Sciences (BioSystems)*, vol. **53**:1-3. Elsevier.

[22] L. Kari, E. Winfree and D. Gifford (eds.), 5th DIMACS workshop on DNA Computers. Preliminary Proc. of the 5th workshop, MIT, 1999.

[23] J. SantaLucia, Jr., A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics *Proc. Natl. Acad. Sci.* **95**, 1460 (1998).

[24] M. Sipper., *Evolution of Parallel Cellular Machines* (The Cellular Programming Approach). Springer-Verlag, Berlin.

[25] W. D. Smith, *DNA Based Computers, Princeton University, 1996*, DIMACS Proc. Series (American Mathematical Society, Providence, RI, 1996).

[26] J. G. Wetmur, *Preliminary Proceedings of the Third Annual Meeting on DNA Based Computers, University of Pennsylvania, 1997*, DIMACS Proc. Series (American Mathematical Society, Providence, RI, 1997).

[27] *Proc. of The Genetic and Evolutionary Computation Conference GECCO-00*, D. Whitley, D. Goldebrg, E. Cantu-Paz, L. Spector, I. Parmee, H.G. Beyer (eds.), Las Vegas, 2000, Morgan Kaufmann.

[28] B-T Zhang, S-Y Shin, Molecular Algorithms for Efficient and Realiable DNA Computing, In [16], 735-742.

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.

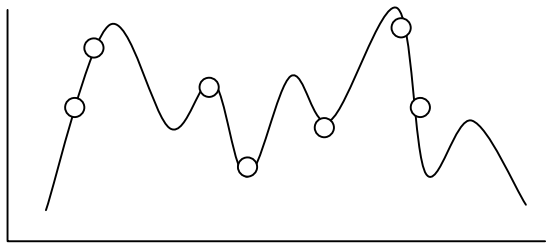

Figure 1: Fitness landscape with individuals

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)

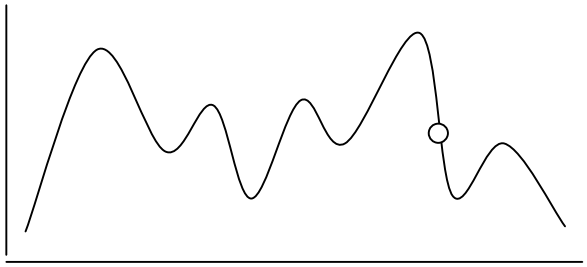

Figure 2: A classical individual is selected

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.

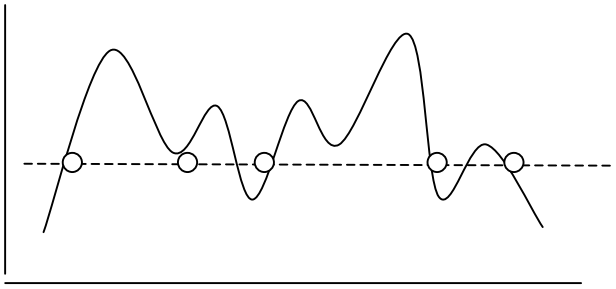

Figure 3: A quantum individual is actually several classical individuals (all those with the same fitness)

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.

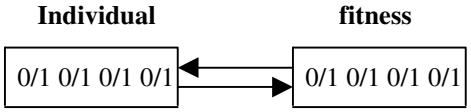

Figure 4: Two 4-qubit entangled quantum registers (representing a single individual)

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)

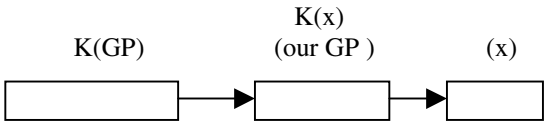

Figure 5: Depiction of the Kolmogorov complexity of a GP and its output

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