Finding Solutions to NP Problems: Philosophical Differences Between Quantum and Evolutionary Search Algorithms

G. W. Greenwood
Dept. of Electrical & Computer Engineering
Portland State University
Portland, OR 97207
greenwood@ieee.org

Published in Proceedings CEC2001, 815-822, 2001

Abstract- This paper uses instances of SAT, 3SAT and TSP to describe how evolutionary search (running on a classical computer) differs from quantum search (running on a quantum computer) for solving NP problems.

1 Introduction

The real challenge of combinatorial optimization is to create algorithms and techniques that can solve realistically sized problems within a reasonable amount of computational time. Most of these algorithms formulate a combinatorial problem as a search problem—i.e., the solutions to combinatorial problems reside in an abstract solution space and two solutions are neighbors if they differ by a single mutation of a problem parameter. Any algorithm that "solves" a combinatorial problem is therefore a search algorithm that explores the solution space landscape.

Unfortunately, many real-world combinatorial problems require such huge computational resources that brute force search methods are useless; they simply take too much time to find the optimal answer. This has led researchers to use search heuristics that yield an acceptable compromise: a possibly lower quality answer but with a minimal search effort. Evolutionary Computation (EC) techniques are at the forefront of this work and impressive results have been achieved (e.g, see [1, 2]). Recently an entirely new approach has surfaced with potentially enormous consequences. This new approach is called *quantum computing* and it relies on the principles of quantum mechanics to evolve solutions.

It is interesting to compare how a quantum search, running on a quantum computer, differs from an evolutionary search, running on a classical computer. However, the whole point of this comparison is *not* to advocate one method over the other—its purpose is to highlight the radically different philosophical approaches. (Besides, because no one has ever built a quantum computer, there is no way any direct comparison can be made at this time. It is up to the reader to decide which approach holds the most promise.) If nothing else, the reader should come away with an appreciation for the total re-orientation in thinking that quantum search will require.

The paper is organized as follows. Section 2 provides an overview of evolutionary algorithms and quantum computing. Every attempt has been made to make the quantum computing portion a comprehensive tutorial that should be understandable by most computer scientists. Because the focus of this paper is on NP-complete and NP-hard problems, a formal definition of these problem classes is also provided. Section 3 describes how the two algorithm methodologies have been used to solve NP problems, which leads into a formal comparison in Section 4. Finally, Section 5 comments on the future of quantum computing.

2 Background

2.1 Evolutionary Algorithms

All evolutionary algorithms (EAs) share the same basic organization: iterations of competitive selection and random variation. Although there are several varieties of EAs, they are all biologically inspired and generally follow the format depicted in Figure 1.

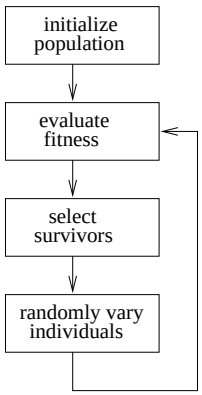

Figure 1: The canonical EA.

The evaluation function for an individual returns a numeric value representing the quality of the solution described by that individual. This numeric value is often called the *fitness* of the individual while the evaluation function is called the *fitness function*. High fitness means the associated individual represents a good solution to the given problem. The selection process targets highly fit individuals for survival. The loop shown in Figure 1 continues until either a fixed number of generations are processed or an acceptable solution has been found.

2.2 Quantum Computing

Almost twenty years ago Richard Feynman observed that classical computers could not simulate certain quantum mechanical effects such as entanglement [3]. This observation spawned interest in the field of quantum computing—i.e., computational machines that perform calculations by emulating quantum mechanic effects. Although no practical quantum computer has yet been built (and the likelihood of building one in the near future is bleak), the interest in this field is growing by leaps and bounds [4]. This section reviews some of the basic concepts of quantum mechanics that relate to quantum computing.

Classical computer systems represent a single bit of information deterministically: the value is either a logic 0 or a logic 1. Quantum computer systems represent a single bit of information as a *qubit*, which is a unit vector in a complex Hilbert space C^2. The ideas are commonly expressed using the bra/ket notation introduced by Dirac [5]. The *ket* symbol is denoted by $|x\rangle$ and the corresponding *bra* is denoted by $\langle x|$. The ket describes a quantum state and the corresponding bra is its complex conjugate.

In computer science domains the ket (bra) can be thought of as a column (row) vector. That is, the orthonormal basis $\{|0\rangle, |1\rangle\}$ can be expressed as $\{(1,0)^{\mathrm{T}}, (0,1)^{\mathrm{T}}\}$. Any complex linear combination of two kets is also a ket. The inner product of two vectors is denoted by $\langle x|y\rangle$. Note that since $|0\rangle$ and $|1\rangle$ are orthonormal, $\langle 0|1\rangle = 0$. $|x\rangle\langle y|$ denotes the outer product of the vectors.

Any practical quantum computer manipulates a register of n qubits. If each qubit has an orthonormal basis $\{|0\rangle, |1\rangle\}$, then a n qubit system has a basis expressed by the *tensor product*: $C^2 \otimes C^2 \otimes \cdots C^2$. This gives 2^n basis vectors of the form

$$\begin{gathered} |0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle \\ |0\rangle \otimes |0\rangle \otimes \cdots \otimes |1\rangle \\ \vdots \\ |1\rangle \otimes |1\rangle \otimes \cdots \otimes |1\rangle \end{gathered}$$

In general, $|a\rangle$ denotes the tensor product $|a_n\rangle \otimes |a_{n-1}\rangle \otimes \cdots \otimes |a_1\rangle \otimes |a_0\rangle$, which means a quantum register has the value $a = 2^0 a_0 + 2^1 a_1 + \cdots + 2^n a_n$.

A qubit need not exist in only one basis state. Indeed, a qubit can exist as a *linear superposition* of basis states $c_0|0\rangle + c_1|1\rangle$, where c_0, c_1 are complex numbers with $|c_0|^2 + |c_1|^2 = 1$. More generally, the n qubit register can be prepared in a superposition of all possible classical states:

$$|x\rangle = \sum_{i=0}^{2^n-1} c_i |i\rangle \tag{1}$$

where the normalization condition $\sum_i c_i^2 = 1$ must hold. The complex number c_i is called the *amplitude* associated with the state $|i\rangle$.

The linear superposition of states is key to understanding how quantum computers operate. This linearity feature means that any operation on a superposition of states renders the superposition of that operation on each state individually [6]. There is no analogue in classical computer system for this principle and, as will be shown below, it is an important ingredient of the power behind quantum computing. However, superposition also permits the following rather bizarre situation. Consider the state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ *This state cannot be expressed in terms of the individual qubit states.* The proof is straightforward. Note that

$$\begin{aligned} &(a_1|0\rangle + b_1|1\rangle) \otimes (a_2|0\rangle + b_2|1\rangle) \\ &\quad = a_1 a_2 |00\rangle + a_1 b_2 |01\rangle + b_1 a_2 |10\rangle + b_1 b_2 |11\rangle \\ &\quad = |00\rangle + |11\rangle \end{aligned}$$

Clearly $a_1 b_2 = 0$, but this implies either $a_1 a_2$ or $b_1 b_2$ must equal zero, which is not possible. States that cannot be described by individual qubit states are called *entangled*. There is considerable debate concerning the actual role entanglement plays in search operations. This issue will be discussed again in Section 3.

The state of a qubit register is determined by a measurement. In quantum systems this measurement process projects the system state onto one of the basis states. Referring to Eq. (1), the measurement returns a value of $|i\rangle$ with probability $|c_i|^2$. Any subsequent measurement returns the state $|i\rangle$ with probability 1, which means the measurement process irreversibly alters the state of the system. Measurement also gives another perspective on entanglement: two qubits are entangled if and only if the measurement of one effects the state of the other.

A quantum computer can perform the same function f as a classical computer if that function is a one-to-one mapping from the domain to the range. In other words, f must be a reversible function. Reversibility is usually mentioned in the context of performing computations without expending heat [7]. Here, however, reversibility must hold or f will not be physically realizable on a quantum computer. Hogg [6] illustrates the importance of reversibility with a simple example. Suppose $f(s_1) = f(s_2) = s_3$. Then for the superposition $|s\rangle = \frac{1}{\sqrt{(2)}}(|s_1\rangle + |s_2\rangle)$ linearity forces $f(|s\rangle) = \frac{1}{\sqrt{(2)}}(|f(s_1)\rangle + |f(s_2)\rangle)$. But this equals $\sqrt{(2)}|s_3\rangle$, which violates the normalization condition.

Quantum systems evolve from state to state according to Schrödinger's equation [8]. Vector states can be expressed as a superposition of basis states each having an amplitude ψ_i. This means evolution occurs by modification of the state amplitudes. Clearly, we would like to increase the amplitude of that state with the desired answer. Suppose we start in state $|a\rangle = \sum \psi_k |a_k\rangle$. This system evolves over time under a linear operator U, i.e., $|a'\rangle = U|a\rangle = \sum \psi'_k |a_k\rangle$. Hence, $\psi' = U\psi$ and the normalization condition is satisfied iff U is unitary. To see this, consider the inner product $(\psi')^\dagger \psi'$, which must equal one because state vectors are orthonormal.

Substituting $\psi' = U\psi$ yields

$$(U\psi)^\dagger(U\psi) \;=\; \psi^\dagger(U^\dagger U)\psi$$

This inner product equals one if $(U^\dagger U) = I$. Hence, U must be unitary.

It is convenient to adopt a simplified notation when describing unitary operations that are applied to individual qubits. Some common unitary operators are

$$\begin{aligned} I: &\quad |0\rangle \rightarrow |0\rangle, \quad |1\rangle \rightarrow |1\rangle \\ X: &\quad |0\rangle \rightarrow |1\rangle, \quad |1\rangle \rightarrow |0\rangle \\ Z: &\quad |0\rangle \rightarrow |0\rangle, \quad |1\rangle \rightarrow -|1\rangle \end{aligned}$$

where I is an identity operator, X a negation operator, and Z a phase shift operator. Suppose we have a 3 qubit register and we want to negate the first qubit and leave the other qubits unaltered. This transformation is denoted by $X \otimes I \otimes I$.

An extremely important transformation is the *Walsh-Hadamard transformation* defined as

$$\begin{aligned} H: &\quad |0\rangle \rightarrow \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ &\quad |1\rangle \rightarrow \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned}$$

When applied to $|0\rangle$, a superposition state is created. When applied to n bits individually, a superposition of all 2^n states is created. Specifically,

$$\begin{aligned} &(H \otimes H \otimes \cdots \otimes H)|000\cdots0\rangle \\ &\quad = \frac{1}{\sqrt{2^n}}((|1\rangle + |0\rangle) \otimes \cdots \otimes (|1\rangle + |0\rangle)) \end{aligned}$$

After distributing the tensor product, this becomes

$$= \frac{1}{\sqrt{2^n}}(|11\cdots11\rangle + \cdots + |00\cdots01\rangle + |00\cdots00\rangle)$$

$$= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \quad ; \quad x \in \{0,1\}^n$$

It is important to emphasize the role superposition plays in quantum computing. Let U_f be a unitary transformation corresponding to a classical function f, i.e., $U_f : |x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$, where $\oplus$ represents bitwise exclusive-or. Notice that this transformation preserves the input—which must be done if f is not invertible—thereby making U_f unitary [9]. We can think of $|y\rangle$ as the hardware of the quantum computer. When this U_f operates on a superposition of states as in Eq. (1), the result is

$$\begin{aligned} U_f\Big(\sum_{i=0}^{2^n-1} c_i|i\rangle|0\rangle\Big) &= \sum_{i=0}^{2^n-1} c_i U_f(|i\rangle|0\rangle) \\ &= \sum_{i=0}^{2^n-1} c_i|i\rangle|0 \oplus f(i)\rangle \\ &= \sum_{i=0}^{2^n-1} c_i|i\rangle|f(i)\rangle \end{aligned}$$

Notice that f is simultaneously applied to all basis vectors. *Hence, a single application of U_f computes all 2^n values of $f(0), \ldots, f(2^n-1)$ at once* [10]. It is this quantum parallelism that is primarily responsible for the enormous interest in quantum computing. But something must be wrong. How can you extract an exponential amount of information out of a linear number of qubits? The answer lies with the amplitudes. If $c_i = c_j \;\forall i,j$, then a measurement will produce any of the 2^n states with equal probability. Furthermore, once that measurement is taken, the system collapses into that measured state and all other information is lost. (You really can't get something for nothing.) Nevertheless, you can exploit this parallelism using the property of *quantum interference*. Interference allows the exponential number of computations performed in parallel to either cancel or enhance each other. Feynman [8] beautifily describes how light waves can constructively or destructively interfere to produce this effect. The goal of any quantum algorithm is to have a similar phenomena occur—i.e., interference increases the amplitudes of computational results we desire and decreases the amplitudes of the remaining results. It is a unitary operator that would alter these amplitudes. Examples of this approach are presented in Section 3.

2.3 NP-Complete vs. NP-Hard Problems

Many papers that discuss NP-complete and NP-hard problems (incorrectly) presume the reader fully understands the difference between these two labels. This paper takes a more formal approach: all terms and three example problems are formally defined. This material is primarily taken from [11]. I begin with the following basic definitions:

Definition 1 *(decision problem)*
A problem in which the only answer is either YES or NO.
Definition 2 *(language)*
The set of all possible input strings to a decision problem that render a YES answer.

The input strings are defined some fixed alphabet of symbols. For example, binary strings are defined over the alphabet $\{0,1\}$.

Definition 3 *(polynomially reducible)*
Let L_1 and L_2 be two languages. L_1 is polynomially reducible to L_2 (denoted by $L_1 \propto L_2$) if there exists some polynomial-algorithm that converts every input instance $i_1 \in L_1$ into another $i_2 \in L_2$.

Reducibility is asymmetric. In other words, if $L_1 \propto L_2$, this does not necessarily mean $L_2 \propto L_1$. Nevertheless, polynomial reducibility does have an important characteristic, which is given in the following theorem:

Theorem: If $L_1 \propto L_2$, and there is a polynomial-time algorithm for L_2, there there is a polynomial-time algorithm for L_1. (See [11], page 343 for proof.)

With these definitions it is now possible to define the algorithm classes P and NP.

Definition 4 *(class P)*
P is the class of languages (decision problems) L that, with input x, can in polynomial time return an answer YES if and only if $x \in L$.
Definition 5 *(class NP)*
NP is the class of languages (decision problems) that can be checked for correctness in polynomial time.

Notice that the above definition says nothing about the computational effort required to get that answer—it merely says to verify the correctness of an answer takes only polynomial time. Whether or not P=NP has yet to be determined. It is now possible to formally define NP-hard and NP-complete. It should be emphasized that the two type of problem classes are *not* interchangeable.
Definition 6 *(NP-hard)*
A problem $\mathcal{P}$ is NP-complete if every other problem in NP is polynomially reducible to $\mathcal{P}$
Definition 7 *(NP-complete)*
A problem $\mathcal{P}$ is NP-complete if (1) $\mathcal{P}$ $\in$NP, and (2) every other problem in NP is polynomially reducible to $\mathcal{P}$

NP-complete problems are decision problems. NP-hard problems ask for the optimal solution to an NP-complete problem. And, they have at least the same level of difficulty to solve as does the corresponding NP-complete problem. There are a very large number of problems that have been proven to be NP-complete.

3 Search Approaches

I can now define the NP problems that are used to compare the two search techniques. Let Σ be a Boolean expression in Conjunctive Normal Form (CNF)—i.e., Σ is the logical *and* of two or more clauses where each clause is the logical *or* of Boolean variables or their complements. An example is $\Sigma = (x + y + \overline{z}) \cdot (\overline{x} + \overline{y}) \cdot (\overline{y} + z)$. This Boolean expression is considered *satisfied* if an assignment of 0s and 1s to the Boolean variables makes Σ equal to a logic 1. The **SATISFIABILITY PROBLEM (SAT)** takes as input a Boolean expression in CNF and asks if there exist an assignment of 0s and 1s to the variables such that the expression is satisfied. A related problem is **3SAT**. This a satisfiability problem in which each clause has exactly three variables. The **TRAVELING SALESMAN PROBLEM (TSP)** is a well known combinatorial problem where the objective is to find a tour of minimal length that visits all N cities with no city visited more than once. This problem is known to be NP-Hard [12].

There is a clear distinction between NP-Complete and NP-Hard: the former is in class NP and the latter is not. Both **SAT** are **3SAT** are NP-Complete, but **TSP** can be either one depending on how it is formulated. If a **TSP** problem asks "does a tour exist of length $\leq k$?", then this is NP-complete because the answer is quickly verified, which makes it in class NP. However, if the problem asks "what is the minimum length tour?", then the problem is only NP-hard. (The answer is not verifiable in polynomial time because this would require a pairwise comparison among all $N!$ possible tours.)

3.1 Evolutionary Search

Although a number of papers have appeared discussing attacking **SAT** problems using EAs, I will focus on the recent work by Bäck, et al [1]. They used an evolution strategy to find solutions to instances of the **3SAT** problem.

The search for a satisfiable solution is difficult because, as the authors point out, the fitness landscape is extremely flat—any genetic search reverts to a random search. Moreover, this type of landscape makes it difficult to define fitness in a meaningful way. The authors get around this situation by adapting a method suggested by a colleague [13]. This alternative method replaces each literal with x with $(y-1)^2$ and $\overline{x}$ with $(y+1)^2$. Furthermore, each conjunction $\wedge$ is replaced by an arithmetic + (sum) and each disjunction $\vee$ is replaced by an arithmetic $\cdot$ (product). The resulting fitness function has a minimum of 0, when the y_i's converge to 1 (true) or -1 (false). Positive values were rounded to 1, and negative values to -1 to check if an individual in the population is a solution. These changes convert **3SAT** into a real-parameter optimization problem, which evolution strategies are ideally suited for.

The evolution strategy randomly initialized the object parameters to values between -1.0 and 1.0. A (15,100)-ES with one standard deviation (σ) was used; σ had an upper limit of 3.0. Later versions introduced various forms of recombination, which ultimately was shown to render the best performing version.

3.2 Quantum Search

Quantum search approaches differentiate between *structured* problems, where partial solutions can be extended to complete solutions, and *unstructured* problems. The unstructured approach can be used for finding solutions for NP-hard problems.

3.2.1 NP-Complete Problems

Ohya and Masuda [14] developed a quantum search method for solving instances of **SAT**. Their algorithm starts with the quantum system in the state

$$|s\rangle \;=\; \frac{1}{\sqrt{2^n}} \sum_{x_1,\ldots,x_n=0}^{1} \otimes_{j=1}^{n} |x_j\rangle \otimes_1^l |0\rangle \otimes |0\rangle$$

for a **SAT** instance with Boolean variables $x_1, \ldots, x_n$. The l qubits are garbage bits needed by reversible logic gates and the least significant qubit (initialized to $|0\rangle$) indicates if the expression is satisfied. Then, using a unitary operator U_f,

$$\begin{aligned} |t\rangle &= U_f\,|s\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x_1,\ldots,x_n=0}^{1} U_f \otimes_{j=1}^{n} |x_j\rangle \otimes_1^l |0\rangle \otimes |0\rangle \end{aligned}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{x_1,\ldots,x_n=0}^{1} \otimes_{j=1}^{n} |x_j\rangle \otimes_{m=1}^{l} |y_m\rangle \otimes |f(\cdot)\rangle$$

where $f(\cdot)$ is the Boolean expression and $y(\cdot)$ are the garbage bits produced during the quantum computation. (Specifically, these are bits used to support reversible computations performed by the reversible logic gates [15].)

The quantum computation is performed by a unitary gate composed of primitive reversible logic gates: a **Not** gate, a **Controlled-Not**, and a **Controlled-Controlled-Not** gate. These logic gate functions are described by the following unitary matrices:

$$\begin{aligned} U_{NOT} &= |0\rangle\langle 1| + |1\rangle\langle 0| \\ U_{CN} &= |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes (|0\rangle\langle 1| + |1\rangle\langle 0|) \\ U_{CCN} &= |0\rangle\langle 0| \otimes |0\rangle\langle 0| \otimes I + |0\rangle\langle 0| \otimes |1\rangle\langle 1| \otimes I \\ &\quad + |1\rangle\langle 1| \otimes (|0\rangle\langle 0| \otimes |1\rangle\langle 0|) \end{aligned}$$

A **SAT** instance with n Boolean variables and m clauses will require $3mn$ steps for truth assignment and substitution. Hence, the unitary operator becomes

$$U_f = U_{3mn} \cdots U_2 U_1$$

where each U_k is a combination of the $U_{NOT}, U_{CN},$ and U_{CCN} unitary matrices. (Details for constructing the $3mn$ U_k matrices are omitted for brevity. Interested readers are referred to [14] where the entire procedure is described.)

A measurement of $|t\rangle$ causes its collapse into a single state and the least significant bit indicates if that state satisfies $f(\cdot)$. This is done by applying a projector $P = \otimes_1^{n+l} I \otimes |1\rangle\langle 1|$ to $|t\rangle$. That is, $f(\cdot)$ is satisfied if $P|t\rangle$ exists, or equivalently $\langle t|P|t\rangle \neq 0$. If out of the 2^n possible solutions there are r solutions that satisfy f, then the probability of measuring a solution is $|P|t\rangle|^2 = r/2^n$. For small r this probability is quite small. *Hence, in practice quantum search algorithms try to exploit quantum interference to amplify the amplitude of the desirable solutions and attenuate all other amplitudes.*

Cerf, et al. [16] provide a good description of exactly how this is done. Their approach relies on an "oracle" function $c(x)$ that equals one for the optimal input x (and zero elsewhere). The goal is for the quantum system to evolve from an initial state $|s\rangle$ to the target state $|t\rangle$ in minimum time. Note that $c(x) = 1$ only at $x = t$. More precisely, the goal is to increase the amplitude of $|t\rangle$ to a point where a measurement will render $|t\rangle$ with the highest probability.

Assume an arbitrary unitary operator U has been found that connects $|s\rangle$ to $|t\rangle$—i.e., $\langle t|U|s\rangle \neq 0$. The probability $|t\rangle$ is actually found is $|\langle t|U|s\rangle|^2$, which means the experiment must be repeated $|\langle t|U|s\rangle|^{-2}$ times on average to guarantee success. However, it is possible to reduce this to the order of $|\langle t|U|s\rangle|^{-1}$—which can be a considerable savings—with an appropriate quantum search algorithm.

The algorithm begins in a superposition of states and any measurement is postponed until the end. Cerf, et al. [16] defined a specific unitary operator $Q = -Ue^{i\pi P_s}U^\dagger e^{i\pi P_t}$ where $P_s = |s\rangle\langle s|$ and $P_t = |t\rangle\langle t|$ are projection operators on $|s\rangle$ and $|t\rangle$. These exponential operators simple flip the phase on a state. For example, the phase of state $|x\rangle$ is flipped by $e^{i\pi P_s}$ iff $x = s$. Since the objective is search for state $|t\rangle$, the oracle is used to implement its exponential operator. That is, $e^{i\pi P_t}|x\rangle = (-1)^{c(x)}|x\rangle$. Then, by repeatedly applying Q, the amplitude of $|t\rangle$ is amplified, beginning at $U|s\rangle$. This amplitude amplification is achieved by the repeated application of Q which, in effect, rotates the starting state $|s\rangle$ into the target state $|t\rangle$. In other words, the beginning state $U|s\rangle$ is rotated to the target state $|t\rangle$ by repeated applications of Q, followed by a measurement. Recall U was an *arbitrary* unitary operator; by using structure information it may be possible to find a better U' so that $U'|s\rangle$ has larger amplitudes in states which are more probable to be solutions. Cerf, et al. [16] describe a method that constructs such a U'.

Grover's quantum search algorithm searches a random database of N items in $O(\sqrt{(N)})$ steps [17]. This means unstructured NP-complete problems can be solved by forming a database of all possible candidate solutions, and then use Grover's algorithm to find the solution. Although this is a considerable speedup over classical machines, it may not be all that impressive. For instance, if one has to find an assignment of one of k values to n total variables, a classical algorithm would take $O(k^n)$ steps while quantum algorithms would still take $O(k^{n/2})$ steps. Nevertheless, the algorithm does find a use with both NP-complete and NP-hard problems.

3.2.2 NP-Hard Problems

Most attempts at solving **TSP** instances with EAs have concentrated on discovering useful recombination operators that try to use partial tour information from the parents to construct good tours in the offspring. Local search operations often augment the main search effort to find even better tours. (These are called *memetic algorithms*.) Unfortunately, the computation time to solve large problems can often take hours.

Tao and Michalewicz [2] proposed a novel EA that considerably reduces the computation effort. In their EA each parent competes for survival only with its offspring. They defined a new unary reproduction operator called "inver-over", which is adaptive: its properties incorporate knowledge taken from the current population. For example, suppose an individual in the current population enumerates the tour

$$S = \{1\ 3\ 8\ 6\ 2\ 5\ 4\ 7\}$$

An offspring is created by randomly choosing two cities (say $c = 3$ and $c' = 5$) and "inverting" the list of intermediate cities. That is, inverting S yields

$$S' = \{1\ 3\ 5\ 2\ 6\ 8\ 4\ 7\}$$

Note that the inversion process makes c and c' adjacent. The inver-over operator chooses c' from another randomly cho-

sen individual in the current population. Specifically, c is located in the randomly chosen individual and its adjacent city becomes c' in the original individual. Evaluation of the offspring does not occur until a sequence of inver-over operators have been applied. This sequence terminates when the c' chosen from a random individual is already adjacent to c in the original individual. Test results indicate this EA can find near-optimal solutions for **TSP** instances with over 400 cities within a few minutes.

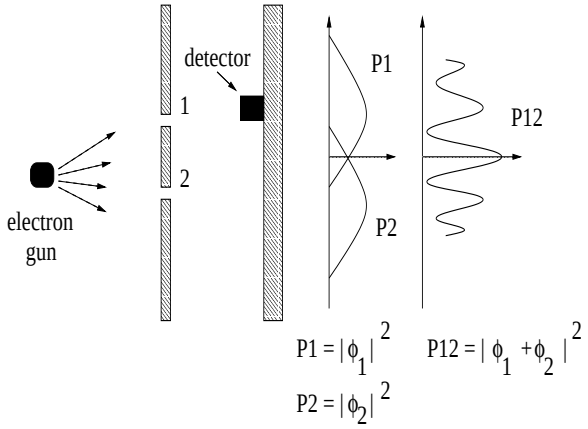

Figure 2: An interference experiment showing how amplitudes combine in both a constructive and destructive manner. The electron gun fires electrons that go through slits in a wall. A movable detector determines where the electrons impact the backstop. The wavelike behavior of the electrons produces interference so that the total distribution $P_{12} \neq P_1 + P_2$. "ϕ" is a complex number called a *probability amplitude*. This figure was adopted from [8].

A beautiful example of how non-traditional architectures can solve NP-hard problems is the scheme presented by Černý [18] to solve an instance of **TSP**. His proposed quantum computer is based on the classical interference experiment, which is shown in Figure 2. The computer has (n-1) walls representing cites 2,3,. . . , n. Furthermore, each wall has (n-1) slits. A beam of quanta (e.g., electrons) sent through this array has (n-1)$^{n-1}$ possible trajectories. The wavelike behavior of electrons means a superposition of all possible trajectories is rendered in $O(n)$ time. A sample trajectory in this quantum computer is shown in Figure 3.

These trajectories identify tours but they do not indicate the length of those tours. Since this machine is hypothetical, an internal degree of freedom can be added—even if nature doesn't provides it. Specifically, the internal state of a particle is $|k; c_2, c_3, \ldots, c_n; p\rangle$ where $k \in \{0, 1, 2, \ldots, NL\}$, $c_i \in \{0, 1\}$, and $p \in \{0, 1\}$. The quantum number k measures the tour length; $c_i = 0$ if city i is not visited and 1 otherwise; and the quantum number p is used to control the dynamics of the search [1].

[1] See the appendix in [18] for an explanation of how p is used; its purpose is not needed for the brief overview given in this paper.

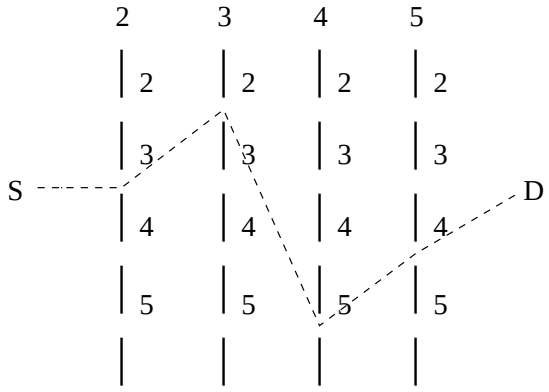

Figure 3: A sample TSP trajectory in the Černý quantum computer. This tour is $S, 3, 2, 5, 4, D$. Note that not all tours are "legal". For instance, $S, 3, 3, 5, 4, D$ is also a trajectory but it is illegal because city 3 is visited twice and city 2 is never visited.

To illustrate the dynamics, let $(i, m) \rightarrow (i + 1, n)$ denote a trajectory between slots on neighboring layers i and $i + 1$ indicating the tour moves from city m to city n. If the particle moves through slot (i, n), then $c_n = 0 \rightarrow c_n = 1$. Furthermore, assume the particle moving between layers encounters a field that increases the quantum number k by a factor d_{mn} if the trajectory moves from (i, m) to $(i + 1, n)$, where d_{mn} is the distance between the two cites. Then, with an initial state $|0; 0, 0, \ldots, 0; 0\rangle$, after passing through the machine the particles are in a state

$$\sum_{\text{trajectories}} |k; c_2, c_3, \ldots, c_n; p\rangle_{\text{trajectory}}. \qquad (2)$$

Note that the legal tours will have all $c_i = 1$ and the quantum number k is the tour length. A filter installed at point D purges all kets with at least one $c_i = 0$. This leaves

$$\sum_{\text{TS routes}} |k; c_2, c_3, \ldots, c_n; p\rangle_{\text{TS route}}. \qquad (3)$$

A Stern-Gerlach-like device [8] could be used to construct such a filter, which would split the above superposition into NL streams according to the k value. A set of particle detectors would then indicate the tour length—i.e., a detector measuring M would fire if there exists a TS tour with a length equal to M.

It is important to emphasize this does *not* mean an instance of **TSP** can be solved in polynomial time. In principle, Eq. (3) could represent a superposition of $O(n!)$ states. Hence, even if Grover's algorithm is used, it would take $O(\sqrt{n!})$ steps to find the minimal length tour.

4 Discussion

No one has yet built a quantum computer capable of searching for solutions to even moderate size NP problems. But,

despite our inability to make head-to-head comparisons of evolutionary and quantum searches, it is possible to highlight their primary philosophical difference:

evolutionary search

> The algorithm uses stochastic operations to explore a fitness landscape comprised of all possible solutions.

quantum search

> The algorithm forms a superposition of all possible problem states. A unitary operator alters the amplitudes of each state exploiting interference to maximize the amplitudes of the desired states. A final measurement extracts the solution with a probability equal to the amplitude squared.

EAs must tradeoff *exploration* against *exploitation*. In other words, the EA must carefully decide which regions of the fitness landscape to abandon, because the solutions are found there are poor, without putting much emphasis on regions with good solutions because that would limit the search. The focus of EA research with respect to NP problems is in two areas: (1) identification of appropriate representations of the problem parameters, which ultimately defines the fitness landscape, and (2) creation of effective stochastic reproduction operators that control movement over the fitness landscape.

Quantum search algorithm exploit superposition to produce massive parallelism. One rather contentious debate in this field is the role entanglement plays. On one side of the fence are those who feel entanglement is essential for speedup [19], while on the other side are those who feel it is completely unnecessary [20]. The latter group believes superposition and interference are sufficient to produce speedup. This issue could be resolved if a truly entangled system were available for study. Unfortunately, recent room-temperature liquid-state NMR experiments have failed to produce any entangled states. Still, some researchers feel increasing the number of qubits (currently only around 5) will eventually make entanglement appear [21].

One of the main difficulties in running a quantum computer is they must remain completely isolated from their environment or the state evolution will cease. Furthermore, there is no way of observing what's going on unless a measurement is taken. But taking a measurement process changes the system by causing it to collapse into one of the basis states. Some methods of dealing with this have been proposed [3], but it still remains a thorny issue. Consequently, we can expect running a quantum search will be much more fragile than the running of an evolutionary search on a classical computer.

One final note on unstructured NP-complete problems. The $O(\sqrt{N})$ time for Grover's search algorithm has been proven to be optimal [22]. This has a rather disappointing consequence: if $O(\sqrt{N})$ time is optimal, this may mean quantum computers can't solve NP problems with an exponential speedup. Preskill [23] suggests the real application area may lay outside NP. Quantum system simulation is one example, which was also previously suggested by Feynman [3].

Even if an exponential speedup isn't possible, why can't evolutionary and quantum computing be combined to find really good solutions quickly? Consider a very simple evolutionary algorithm where a single parent competes against its λ offspring for survival. The parent for the next generation is selected with roulette wheel sampling [24]. A quantum version, in principle, could be constructed as follows. Suppose the single parent were represented by n qubits. A Walsh-Hadamard transformation of $k << n$ qubits creates a superposition of the parent and $\lambda = 2^k - 1$ offspring. A unitary transformation would amplify the amplitude of the best offspring, and a final measurement would chose the single parent for the next generation. This might be a reasonable approach for searching extremely large solution spaces—e.g., a n=50 variable **SAT** problem—where manipulating all 2^n possible solutions at one time would be difficult.

5 Final Comments

I will conclude this paper with some personal observations. I do believe quantum computing will change the way computer engineers and scientists think about computing systems. To date, quantum computing has been the domain of primarily physicists. It is about time that computer engineers and scientists enter this arena and begin to drive its direction.

Many computer professionals entering this field are quickly put off by the notational and conceptual barriers. Tutorials are available (e.g., see [15, 25, 26, 27]), but many readers will quickly find them incomprehensible—they are written by physicists for physicists. (Out of this lot, however, I believe [15] is the best.) The sad truth is a computer professional who lacks a firm foundation—i.e., formal training—in quantum theory will most likely not be able to contribute to the quantum computing field. As an absolute minimum I would recommend an upper division undergraduate course in quantum mechanics. This should be sufficient background for one to begin reading the literature from the field.

My other observation concerns the practicality of the currently proposed quantum computer architectures. A number of proposed systems advocate a vast network of interconnected quantum gates (such as AND gates) which implements some function $f(x)$. I believe this is entirely too low of a design level, which is unlikely to lead to massive improvements in computation power—certainly no where near orders of magnitude improvement. Although, in principle, all computer systems are just interconnected primitive logic gates, computer engineers don't look at them in this way. Designers do not think of a processor as a network of primitive logic gates manipulating input binary strings. Today's computer systems are simply too complex, which is precisely why dense integrated circuits are normally designed at a register-

transfer level using hardware description languages such as VHDL or Verilog. Moreover, thinking of quantum computers in terms of interconnected logic gates likewise limits their ability to perform general purpose computations—especially those computations that are inherently parallel. For instance, can a quantum computer perform an evolutionary search such as that described in the previous section?

I am convinced that a radical increase in computing power will only come once the Von Neumann paradigm has been dispensed with. Architectures such as those proposed by Černý [18] are an example of the imagination that will be required.

Bibliography

[1] T. Bäck, A. Eiben, and M. Vink. A superior evolutionary algorithm for 3SAT. *Proc. EP98*, 1998.

[2] G. Tao and Z. Michalewicz. *Inver-over Operator for the TSP*. in *Proc. PPSN V*, T. Bäck, A. Eiben, M. Schoenauer, and H.-P. Schwefel (Eds.), 803-812, 1998.

[3] R. Feynman. Simulating physics with computers. *Intl. J. Theo. Phys.*, 21:467–488, 1982.

[4] National Science Foundation. Quantum information science. *Report of the NSF Workshop, Arlington, VA*, 1999.

[5] P. Dirac. *The Principles of Quantum Mechanics*. Oxford University Press, 4th edition, 1958.

[6] T. Hogg. *Quantum computing and phase transitions in combinatorial search*. Los Alamos Physics preprint archive, http://xxx.lanl.gov/abs/quant-ph/9508012, 1996.

[7] C. Bennett. Logical reversibility of computation. *IBM J. Res. Dev.*, 17:525–532, 1973.

[8] R. Feynman, R. Leighton, and M. Sands. *Lectures on Physics, Vol. III*. Addison-Wesley, 1965.

[9] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. Royal Soc. of London A*, A400:97–117, 1985.

[10] A. Barenco. *Quantum computation: an introduction*. in *Introduction to Quantum Computation and Information*, H. Lo, S. Popescu and T. Spiller (Eds.), World Scientific, 1998.

[11] U. Manber. *Introduction to Algorithms*. Addison-Wesley, 1989.

[12] M. Garey and D. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W.H. Freeman & Company, 1979.

[13] Z. Michalewicz. personal communication with authors of [1].

[14] M. Ohya and N. Masuda. *NP problem in quantum algorithm*. Los Alamos Physics preprint archive, http://xxx.lanl.gov/abs/quant-ph/9809075, 1998.

[15] E. Rieffel and W. Polak. *An introduction to quantum computing for non-physicists*. Los Alamos Physics preprint archive, http://xxx.lanl.gov/abs/quant-ph/9809016, 2000.

[16] N. Cerf, L. Grover, and C. Williams. *Nested quantum search and NP-complete problems*. Los Alamos Physics preprint archive, http://xxx.lanl.gov/abs/quant-ph/9806078, 1998.

[17] L. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Phy. Rev. Ltr.*, 79:325–328, 1997.

[18] V. Černý. Quantum computers and intractable (NP-complete) computing problems. *Phys. Rev. A*, 48:116–119, 1993.

[19] S. Braunstein and A. Pati. *Speedup and entanglement in quantum searching*. Los Alamos Physics preprint archive, http://xxx.lanl.gov/abs/quant-ph/0008018, 2000.

[20] P. Knight. Quantum information processing without entanglement. *Science*, 287:441–442, 2000.

[21] R. Fitzgerald. What really gives a quantum computer its power? *Physics Today*, pages 20–22, 2000.

[22] C. Zalka. *Grover's quantum searching algorithm is optimal*. Los Alamos Physics preprint archive, http://xxx.lanl.gov/abs/quant-ph/9711070, 1997.

[23] J. Preskill. *Quantum computing: pro and con*. Los Alamos Physics preprint archive, http://xxx.lanl.gov/abs/quant-ph/9705032, 1997.

[24] J. Grefenstette. *Proportional selection and sampling algorithms*. in *Evolutionary Computation 1: Basic Algorithms and Operators*, T. Bäck, D. Fogel, and T. Michalewicz (Eds.), IOP Publish., 172-180, 2000.

[25] V. Vedral and M. Plenio. *Basics of quantum computation*. Los Alamos Physics preprint archive, http://xxx.lanl.gov/abs/quant-ph/9802065, 1998.

[26] D. Aharonov. *Quantum computation*. Los Alamos Physics preprint archive, http://xxx.lanl.gov/abs/quant-ph/9812037, 1998.

[27] Jr. S. Lomonaco. *A rosetta stone for quantum mechanics with an introduction to quantum computation*. Los Alamos Physics preprint archive, http://xxx.lanl.gov/abs/quant-ph/0007045, 2000.