Quantum Computing Applied to Optimization

Yuzhen Ge[†] Layne T. Watson[‡]

Abstract

Optimization problems represent a class of problems that can be time consuming to solve and very complex. In this paper, a quantum algorithm for solving optimization problems is proposed. The algorithm utilizes the encoding scheme from genetic algorithms to encode the problem and then uses Grover's unitary transformation to seek out a solution. The efficiency of the algorithm depends on the length of the chromosome or the coded variables. As a simple example the satisfiability problem, a NP-complete problem, is examined using the algorithm and the time complexity of solving this problem is greatly improved. The traveling salesman and minimum spanning tree problems are also briefly discussed.

1 Introduction

The idea of quantum computing, a completely new paradigm for computation, was first proposed by Feynman [7] [8] more than a decade ago. It had not received much attention until Shor showed the existence of a polynomial time algorithm on a quantum computer to solve the discrete log problem and the factoring problem [13]. Even though whether a quantum computer can be built is still questionable, the fact that factoring problems can be solved in polynomial time is profound. Various work on the capability of quantum computers has appeared in [1], [3], [4], [5], [12], [14].

Many papers have explored the possibilities on a quantum computer. One of them in particular, due to Grover[11], shows that a search problem with unsorted lists can be solved in $O(\sqrt{N})$ steps on a quantum computer compared to $O(N)$ steps on a standard computer. However, in Grover's work, no detail about mapping the unsorted list to quantum states is given. If this step is not omitted, unless a smart mapping scheme is developed, the time complexity of the algorithm will likely be $O(N)$ instead of $O(\sqrt{N})$, since the list to state mapping is (apparently) linear. The proposed quantum algorithm uses Grover's unitary transformation to solve an optimization problem, but with a different preprocessing of the data.

While the techniques in Grover's work are very clever, Grover's basic idea is relatively straightforward: apply a unitary transformation so that the amplitude of the state matching the solution will increase. Grover's techniques and ideas are used to investigate possible algorithms to solve an optimization problem on a quantum computer.

Optimization problems are encountered in every discipline that involves mathematics. Some optimization problems (e.g., NP hard discrete combinatorial problems) are extremely challenging to solve on a standard computer. There are many different algorithms for solving various kinds of optimization problems, including many heuristic methods, genetic algorithms being one such heuristic method.

The work of Layne T. Watson was supported in part by Air Force Office of Scientific Research grant F49620-96-1-0104 and National Science Foundation grant DMS-9625968.

† Department of Computer Science, University of the Pacific, Stockton, CA 95219.

‡ Departments of Computer Science and Mathematics, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061-0106.

The capability of quantum computers is discussed in other publications. One of the advantages of quantum computer is that all different states can coexist and a solution state can be sorted out by incrementing its amplitude while suppressing that of other states. As a result, some optimization problems may be solved efficiently.

A genetic algorithm [2] consists of computation rules that model the process of natural evolution. Organisms go through a process of mating (crossover), mutation, and selection, with the fittest individuals surviving to create new offspring from generation to generation. From a science and engineering standpoint, a genetic algorithm is simply a stochastic search technique based on the principles of genetics for solving an optimization problem. The terminology of the literature also mirrors that of genetics.

A subset of all feasible points for an optimization problem will be called a population. Each individual (feasible point) in a population will be represented as a single chromosome, which is a string of genes (in general, individuals may be encoded as several chromosomes). A gene is generally an integer or a binary bit. For each chromosome (individual), a fitness can be computed based on its gene string and the objective for optimization.

Once the initial population is chosen, crossover, mutation, and then selection are performed to generate new chromosomes in a population. The reason that these operations are performed is that it is too time consuming to treat all possible chromosomes and therefore only a small portion of the feasible set is kept in a population. However, to explore more of the feasible set, these operations are used to generate other chromosomes not in the population, then a selection mechanism is chosen to keep the "best" chromosomes in the population. One early attempt was to simply emulate genetic algorithms on a quantum computer [9]. Simple emulation of genetic algorithms as executed in traditional computers will not exploit quantum computing advantages. A different approach must be adopted to utilize the special properties of quantum computers.

2 The Algorithm

In a quantum computer, if we code all the feasible points as in a genetic algorithm, we can include all of them in a superposition of states simultaneously. Instead of performing operations like crossover, mutation, selection, etc. to select the "fittest" one, unitary transformations are performed to single out the chromosome or state that corresponds to the optimizing point in a quantum computer.

The details of the algorithm follow.

1. Encode the problem. This step is similar to the first step required in a genetic algorithm. Various methods developed in genetic algorithms can be used here. In this step, an optimization problem is transformed so that the feasible points are discretized and represented by finite length bit strings.

2. Prepare for quantum computation.

If n binary bits are needed to encode a chromosome, we will use a register of n qubits to represent all the possible states of the quantum machine or the possible points in the feasible region of the optimization problem. We prepare the initial state to be a pure state such that every possible chromosome will have the same probability in the state. If there are $N = 2^n$ states, then the initial state is

$$\frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \, .$$

In other words, all the points in the feasible region of the optimization problem are involved in the initial state.

As mentioned in [11], $O(\sqrt{N})$ unitary operations are needed to prepare the initial state.

3. Perform the quantum computation.

```
for i = 1 to √N do
{
  If J(S) = 0, rotate the phase of the state by π radians.
  Apply unitary matrix D (diffusion transform).
}
```

where D is defined as

$$D_{ij} = \frac{2}{N} \qquad \text{if } i \neq j$$

and

$$D_{ii} = -1 + \frac{2}{N},$$

and $J(S) = 0$ corresponds to some optimization criterion for the problem, which may be different for different types of optimization problems. One of the difficulties in using this method may be the formulation of such a criterion. For our example of the satisfiability problem, it is straightforward.

Rotation by π radians can be done by the unitary transformation $e^{j\pi}$. The diffusion transformation D will increase the amplitude of the states that were rotated by an angle π earlier as shown in [11].

4. Measure the final state.

As shown by Grover, after $O(\sqrt{N})$ iterations of the loop in Step 3 the probability of the desired state will reach $O(1)$.

3 The Satisfiability Problem

One famous NP-complete problem is the satisfiability problem. Cook [6] proved that the satisfiability problem is NP-complete. Let E be a Boolean expression over the variables $x_1, x_2, \cdots, x_n$ in conjunctive normal form (CNF), i.e., a product (AND) of sums (ORs). For example, E can be

$$E = (x_1 + x_2 + \bar{x}_3) \cdot (\bar{x}_1 + x_2 + x_3) \cdot (\bar{x}_1 + \bar{x}_2 + \bar{x}_3),$$

where x_i or $\bar{x}_i$ (NOT x_i) is called a *literal* and each sum above is called a *clause.*

The satisfiability problem can be stated as follows:

Instance: A set of *literals* $X = \{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_n, \bar{x}_n\}$ and a set of *clauses* $C = \{c_1, c_2, \ldots, c_m\}$, where each clause c_i is a subset of X.

Answer: "Yes" if there is a (satisfying) assignment of values of the variables $\{x_1, x_2, \ldots, x_n\}$ such that each clause has at least one literal whose value is 1; in this case, the boolean expression $E(C) = c_1 \cdot c_2 \cdots c_n = 1$.

For example, for E above, $C = \{c_1, c_2, c_3\}$, where $c_1 = \{x_1, x_2, \bar{x}_3\} \leftrightarrow x_1 + x_2 + \bar{x}_3$, $c_2 = \{\bar{x}_1, x_2, x_3\} \leftrightarrow \bar{x}_1 + x_2 + x_3$, and $c_3 = \{\bar{x}_1, \bar{x}_2, \bar{x}_3\} \leftrightarrow \bar{x}_1 + \bar{x}_2 + \bar{x}_3$. The satisfiability problem is trying to answer the question: is $E = c_1 \cdot c_2 \cdot c_3$ satisfiable? That is, is there a truth assignment for x_1, x_2, x_3 such that each c_i, $1 \leq i \leq 3$, is true or E is true? One assignment that will make E satisfiable is $x_1 = 1$, $x_2 = 1$, and $x_3 = 0$.

Satisfiability, stated as an optimization problem, is

$$\max_{x_i \in \{0,1\}} E(C).$$

Note that $\max E(C) = 1$ if and only if $E(C)$ is satisfiable.

In a traditional computer, the average time to find a satisfying assignment of values $\{x_1, x_2, \cdots, x_n\}$ is of order $O(2^n/2 = 2^{n-1})$. We will use the quantum algorithm to solve this problem. Since the variables are discretized, there is no need to use a genetic algorithm encoding to discretize the variables. n qubits will be needed for quantum computation. Assume that there is only one truth assignment that satisfies E. Since there are $N = 2^n$ possible states, the time complexity is of order $O(\sqrt{N}) = O(2^{\frac{n}{2}}) << O(2^{n-1})$ when n is large, a great improvement over any known classical algorithm.

The algorithm can be described as follows:

1. Preparation for quantum computation — since n qubits are needed for the n Boolean variables, there are $N = 2^n$ possible states. The initial state is prepared to be a pure state such that every possible state will have the same probability in the superposition, that is, the initial state is

$$\frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle .$$

2. Perform the quantum computation

for $i = 1$ to $\sqrt{N}$ do

{

if $E(C) = 1$, rotate the phase of the state by π radians.

Apply unitary matrix D

}

where D is defined as

$$D_{ij} = \frac{2}{N} \qquad \text{if } i \neq j$$

and

$$D_{ii} = -1 + \frac{2}{N}$$

Step 3. Observe, measure, and check the final state.

In Step 3, the final state needs to be checked to verify that it is a solution for the problem. If it is not, than the problem does not have a solution. In the situation that there is no solution for the problem, all the states are equally likely to show up as the final state, making the final step necessary.

For a different problem, the criterion in the loop will be different and therefore the quantity $J(S)$ has to be formulated and defined before this method can be applied. Another example, the modified traveling salesman problem, will be examined. This problem involves genetic algorithm encoding techniques.

4 The Modified Traveling Salesman Problem

The *traveling salesman problem* is a classic NP-complete problem in combinatorial optimization and is one of the most widely studied problems. The *modified traveling salesman problem* asks if there is a tour that is shorter than T_0? It is not difficult to see that it is also as hard as the original traveling salesman problem and the time complexity is of order $O(n!)$, where n is the number of cities in the tour.

There are existing genetic algorithms to solve the traveling salesman problem. What we need to use is an encoding algorithm that will encode a tour as a state in a quantum computer and there are a total of $N = (n-1)!$ states available. According to Section 2, the performance will be of order $O(\sqrt{N})$.

5 The Modified Minimum Spanning Tree Problem

The *minimum spanning tree* problem is an NP-hard problem. Most of its extensions are generally NP-hard. This problem has many applications in telecommunication network design, transportation problems, etc. The problem is stated in terms of a connected and undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is a finite set of vertices and E is a finite set of unordered pairs of distinct vertices from V. Elements $e_{ij} = \{v_i, v_j\} \in E$ are called edges. Each edge e_{ij} may have a weight (positive real number) w_{ij} associated with it. A spanning tree is a minimal set of edges that connects all the vertices in the graph and a minimum spanning tree is one with minimum total weight. The *modified minimum spanning tree problem* that we will examine is to answer the question: is there a spanning tree with weight less than W_0?

An encoding method from genetic algorithms can be used to encode a tree for the quantum algorithm. Since any chromosome must be a tree, it is critical to encode all the possible trees in an equally likely manner. There are several different encoding schemes for a tree [10]. The one repeated here is based on Cayley's theorem which states that there is a one-to-one correspondence between a tree in a complete graph with n vertices and a string of $n-2$ digits, where each digit has value less than n. For example, for a graph with 6 nodes, only four digits less than 6 are necessary to represent all the trees.

The computation can be carried out in quantum states entirely, as long as the calculation of the weight can be done directly from the string representation. The time complexity will be of order $O(\sqrt{N})$, where N is the number of states required to represent all the trees.

In summary, a quantum algorithm for solving a generic optimization problem is proposed. The satisfiability problem is used to illustrate the algorithm. Since the variables in the satisfiability problem are discrete already, there is no need to use a genetic algorithm encoding to discretize the variables. The time complexity of the quantum algorithm depends on the total number of possible states. If the total number of states is N, the time complexity of the algorithm is of order $\sqrt{N}$.

References

[1] Leonard M. Adleman, Jonathan DeMarrais, Ming-Deh A. Huang , *Quantum Computability*, SIAM Journal on Computing, Vol. 26, No. 5, pp. 1524–1540, 1997.

[2] Thomas Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, 1996.

[3] Adriano Barenco, Artur Ekert, Kalle-Antti Suominen, Paivi Torma, *Approximate Quantum Fourier Transform and Decoherence*, Phys. Rev. A, Vol. 54, No.1, p. 139, 1996.

[4] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, Umesh V. Vazirani, *Strengths and Weaknesses of Quantum Computing*, SIAM Journal on Computing, Vol. 26, No. 5, pp. 1510–1523, 1997.

[5] Isaac L. Chuang, Yoshihisa Yamamoto, *A Simple Quantum Computer*, Phys. Rev. A, Vol. 25, No. 5, p. 3489, 1995.

[6] Stephen A. Cook, *The complexity of theorem proving procedures*, Third Annual ACM Symposium on Theory of Computing, New York, pp. 151–158, 1971.

[7] Richard P. Feynman, *Simulating Physics with Computers*, International Journal of Theoretical Physics, Vol. 21, pp. 467–488, 1982.

[8] Richard P. Feynman, *Quantum Mechanical Computers*, Foundations of Physics, Vol.16, No.6, 1986.

[9] Y. Ge, L. T. Watson, and E. G. Collins, Jr., *Genetic Algorithms for Optimization on a Quantum Computer*, Proceedings of the first International Conference on Unconventional Models of Computation, Auckland, New Zealand, 1998, Springer, pp. 218–227.

[10] Mitsuo Gen and Runwei Cheng, *Genetic Algorithms & Engineering Design*, John Wiley & Sons, New York, 1997.

[11] Lov K. Grover, *Quantum Mechanics Helps in Searching for a Needle in a Haystack*, Phys. Rev. Lett., Vol. 79, pp. 325–328, 1997.

[12] Tad Hogg, *Highly Structured Searches with Quantum Computers*, Phys. Rev. Lett, Vol. 80, No. 11, pp. 2473, 1998.

[13] Peter W. Shor, *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, SIAM Journal on Computing, Vol. 26, No. 5, pp. 1484–1509, 1997.

[14] Daniel R. Simon, *On the Power of Quantum Computation*, SIAM Journal on Computing, Vol. 26, No. 5, pp. 1474–1483, 1997.