An $R \parallel C_{max}$ Quantum Scheduling Algorithm

Feng Lu and Dan C. Marinescu
School of Computer Science
University of Central Florida, P. O. Box 162362
Orlando, FL 32816-2362
Email: (lvfeng, dcm)@cs.ucf.edu

March 6, 2008

Abstract

Many scheduling problems are NP-hard on classical computers. Using quantum parallelism and entanglement, a quantum schedule algorithm may be able to lead to an exponential improvement. The algorithm presented in this paper constructs a superposition of all schedules and a superposition of their makespans and then amplifies the one that corresponds to the solution. We perform $O(\sqrt{2^n+q})$ Grover iterations. The time complexity of the quantum scheduling algorithm for an $R \parallel C_{max}$ problem is $O(\sqrt{2^n+q})$ while the complexity of a classical algorithm is $O(2^{m/2n})$.

1 Introduction

In recent years we have witnessed significant theoretical and some encouraging experimental results in the area of quantum computing. In 1994, Peter Shor found a polynomial time algorithm for the factorization of $n$-bit numbers on quantum computers [24]. His discovery generated a wave of enthusiasm for quantum computing, for two major reasons: the intrinsic intellectual beauty of the algorithm and the fact that efficient integer factorization is a very important practical problem. The security of widely used cryptographic protocols is based upon the conjectured difficulty of the factorization of large integers.

Like most factorization algorithms, Shor’s algorithm reduces the factorization problem to the problem of finding the period of a function, but uses quantum parallelism to find a superposition of all values of the function in one step. Then the algorithm calculates the Quantum Fourier Transform of the function, which sets the amplitudes into multiples of the fundamental frequency, the reciprocal of the period. To factor an integer, Shor’s algorithm measures the period of the function¹.

In 1996, Grover described a quantum algorithm for searching an unsorted database containing $N$ items in a time of order $\sqrt{N}$ while on a classical computer the search requires a time of order $N$ [10]. The critical aspect of a quantum algorithm is to create a superposition of all possible states and amplify the “solution”. The speedup of Grover’s algorithm is achieved by exploiting both quantum parallelism and the fact that in quantum theory a probability is the square of an amplitude. In 1997, Charles Bennett showed that Grover’s algorithm is optimal. No quantum algorithm can solve this problem faster than time of order $\sqrt{N}$.

Quantum fault tolerance and quantum error correcting codes, quantum dots and NMR-related techniques [30], are some of the areas of quantum computing that give us hope that the day when a quantum computer will be able to perform useful tasks is approaching. Yet, the effervescent activity in theoretical and experimental physics related to quantum computing is not mirrored by ¹A powerful version of the technique used by Shor is the phase-estimation algorithm of Kitaev [18]
the development of quantum algorithms, necessary to justify the intellectual and financial investment in quantum computing. In a recent paper [27], Peter Shor classifies the quantum algorithms known to offer a significant speed-up over their classical counterparts into three broad categories:

1. Algorithms that find the periodicity of a function using Fourier Transforms. Simon’s algorithm [28], Shor’s algorithms for factoring and for computing discrete logarithms [25], and Hallgren’s algorithm to solve the Bell equation are all members of this class.

2. Search algorithms which can perform an exhaustive search of $N$ items in $\sqrt{N}$ time. Grover’s algorithms [10, 11, 12] belong to this class.

3. Algorithms for simulating quantum systems, as suggested by Feynman. This is a potentially large class of algorithms, but not many algorithms in this class have been developed so far. Once quantum computers become a reality we should expect the development of a considerable number of programs to simulate quantum systems of interest [20, 22].

Clearly, the development of quantum algorithms requires a very different thinking than the one we are accustomed to in the case of classical computers [3, 4, 5, 6, 21]. To offer a considerable speed-up, a quantum algorithm must rely on superposition states and this is a foreign concept for those unaccustomed to quantum mechanical concepts [13, 16, 17, 26].

In addition to this obvious reason, Peter Shor speculates [27] that the number of problems the quantum algorithms may offer a substantial speedup over the classical algorithms may be very limited. To see the spectacular speedups we have to concentrate on problems not in the classical computational class $\mathcal{P}$. Many believe that quantum algorithms solving $\mathcal{NP}$-complete problems in polynomial time do not exist, even though no proof of this assertion is available at this time. Shor argues that if we assume that no polynomial time quantum algorithms exist for solving $\mathcal{NP}$-hard problems, then the class of problems we have to search for is neither $\mathcal{NP}$-hard, nor $\mathcal{P}$, and the population of this class is relatively small.

The algorithms used to simulate quantum mechanical systems follow the evolution in time of a fairly large number of quantum particles. The result of such a simulation reflects what the outcome of a measurement of the physical quantities of the quantum system would reveal.

There is no simple algorithmic formulation for the simulation of quantum mechanical systems. Moreover, some of the physical quantities of interest are not known to be accessible even on a quantum computer [19]. For example, the question of what is the ground energy state of a quantum system whose Hamiltonian is known does not have an easy answer. Recall that the Hamiltonian is the operator corresponding to the energy observable of a quantum system. Finding a simple solution to the problem of determining the ground energy state of a quantum system would lead to efficient algorithms for “hard” computational problems such as optimal scheduling, or the well-known travelling salesman problem.

Classical Monte Carlo algorithms are widely used today to simulate quantum physical systems. In Monte Carlo simulations, state amplitudes are represented by the expected values of random variables calculated during the simulation. Such simulations produce large statistical errors that can only be reduced by repeating the calculations many times. The advantage of using quantum algorithms for simulations of quantum physical systems is that they allow us to determine the value of relevant physical properties with polynomial bounded statistical errors.

Many scheduling problems are $\mathcal{NP}$-hard on classical computers [23]. Using quantum parallelism and entanglement, a quantum schedule algorithm may be able to allow us to solve scheduling problems in polynomial time.

In the next sections we discuss a solution for a $R||C_{max}$ scheduling problem. This problem can be formulated as follows: given a set of $N = 2^n$ jobs $J_i$, $1 \leq i \leq 2^n$, a set of $M = 2^m$ machines, $M_j$, $1 \leq j \leq 2^m$ and the execution time of all the jobs on all the machines, $1 \leq p_{ij} \leq 2^m$, does a schedule $\mathcal{A}$ with the makespan $C_{max}^\mathcal{A} = \mu$ exist, and if it does, what is the schedule $\mathcal{A}$.

The algorithm discussed in this paper makes extensive use of a generalized version of Grover’s search algorithm. Given a range of makespans $\mu_{low} \leq \mu \leq \mu_{high}$ we can use the algorithm repeatedly to find a schedule with a makespan in that range. In the design of real-time systems we are often
faced with the problem of finding a schedule able to meet a deadline in a given interval; the algorithm presented in this paper could be applied to solve such problems.

In Sections 2 and 3 we introduce scheduling algorithm and then we discuss information encoding for the \( R|C_{\text{max}} \) scheduling problem. Sections 4 and 7 provide some background material on quantum measurements and on a generalized Grover’s search algorithm.

## 2 Scheduling Algorithms

Scheduling is the problem of assigning tasks to a set of resources subject to a set of constraints, over time, in an “optimal” manner.

We are given a set of \( N = 2^n \) jobs, \( J = \{J_1, J_2, \ldots, J_N\} \) and a set of \( M = 2^m \) machines, \( M = \{M_1, M_2, \ldots, M_M\} \). A schedule \( A \) for the sets \((J, M)\) specifies which \( p_{ij} \) units of time machine \( M_j \) uses to process job \( J_i \). We call \( C_i^A \) the completion time of job \( J_i \) under the schedule \( A \). The makespan of the schedule \( A \) is the maximum completion time of any job in schedule \( A \):

\[
C_{\text{max}}^A = \max_i C_i^A.
\]

A scheduling problem is characterized by a tuple \((\alpha | \beta | \gamma)\) where \( \alpha \) denotes the machine environment, \( \beta \) summarizes the set of constraints, and \( \gamma \) denotes the optimality criterion.

We are often interested in scheduling problem involving multiple machines. We distinguish three cases in the parallel machine environment:

1. **Identical parallel environment.** All the machines are identical and job \( J_i \) requires \( p_i \) units of time on any machine,

2. **Uniformly related parallel environment.** Each machine \( M_j \) has a speed \( s_j > 0 \) and job \( J_i \) if processed entirely on machine \( M_j \) would take \( p_i / s_j \) units of time, and

3. **Unrelated parallel machine environment.** Different machines have different capabilities and the speed of machine \( M_j \) on job \( J_i \) is \( s_{ij} \). Then the processing time of job \( J_i \) on machine \( M_j \) is \( p_{ij} = p_i / s_{ij}, 0 \leq p_{ij} \leq Q \) and \( Q = 2^q \).

These machine environment are denoted by \( P, Q, \) and \( R \), respectively.

Examples of scheduling constraints include deadlines (e.g., job \( i \) must be completed by time \( t \)), resource capacities (e.g., there are only 5 machines), precedence constraints on the order of tasks (e.g., one must be done before another), and priorities on tasks (e.g., finish job \( j \) as soon as possible while meeting the other deadlines). A priority rule assigns to job \( J_i \) a priority \( \pi_i \). A busy schedule is one when one machine becomes available it starts processing the job with the highest priority.

Each scheduling problem could have problem specific optimization criteria. The one which requires the minimization of the makespan is referred to in scheduling theory as \( C_{\text{max}} \). Other optimization criteria can be considered. For example, we may wish to optimize the average completion time of all jobs:

\[
\frac{1}{N} \sum_{i=1}^{N} C_i^A
\]

an optimization criterion denoted as \( \sum_{i=1}^{N} C_i^A \).

We consider an \( R|C_{\text{max}} \) scheduling problem in which all machines are unrelated. All jobs are available at the beginning time and there are no precedence constraints. As no preemption is allowed, once job \( J_i \) started processing on machine \( M_j \) it must complete its execution before another job, \( J_k \) can be processed on \( M_j \).
Such an \( R||C_{max} \) scheduling problem is \( \mathcal{NP} \)-hard. Given a set of \( N = 2^n \) jobs and \( M = 2^m \) machines we can construct \( 2^{(m2^n)} \) different schedules, thus the time complexity of an \( R||C_{max} \) scheduling problem on a classical computer is \( O(2^{(m2^n)}) \). A quantum scheduling algorithm, which exploits quantum parallelism and entanglement, provides an exponential improvement.

3 Information Encoding for the \( R||C_{max} \) scheduling problem

Now we have a set of \( N = 2^n \) jobs running on \( M = 2^m \) machines. In our model all jobs are available when we start processing and no preemption is allowed, once a job is started, it cannot be interrupted.

We assume that processing times are integers in the range \( 0 \leq p_{ij} < Q = 2^q \), \( 1 \leq i \leq 2^n \), \( 1 \leq j \leq 2^m \), and that the jobs could have different processing times on different machines. We also assume that we have a quantum system with \( r = m + q \) qubits for each job.

Given job \( J_i \) running on machine \( M_j \), we encode the job-machine information as a vector \( | e_{ij} \rangle \) obtained as the tensor product of the machine index, \( j \), and the processing time, \( p_{ij} \):

\[
| e_{ij} \rangle = | j \rangle \otimes | p_{ij} \rangle.
\]

Then we define job state vectors for job \( i \), \( | J_i \rangle \) as any superposition of its job-machine vectors. First, we consider an equal superposition of job-machine vectors as:

\[
| J_i \rangle = \frac{1}{2^{m/2}} \sum_{j=1}^{2^m} | e_{ij} \rangle.
\]

A schedule is the tensor product of job-machine vectors:

\[
| A_k \rangle = \otimes_{i=1}^{2^n} | e_{ij} \rangle, \quad 1 \leq j_i \leq 2^m.
\]

A schedule should include one job-machine vector for each job. A specific machine may be present in multiple job-machine vectors, when the schedule requires multiple jobs to be executed on the same machine, or may not appear in any job machine vectors of the schedule \( | A_k \rangle \) if none of the jobs is scheduled on that machine.

The superposition of all schedules is:

\[
| A \rangle = \frac{1}{\sqrt{\sigma}} \sum_{k=1}^{\sigma} | A_k \rangle, \quad \sigma = 2^{m2^n}.
\]

Let \( \Omega \) be an operator which given a schedule constructs the running time on each machine for that schedule. When applied to a superposition of all schedules it produces a superposition of the running time on each machine for all schedules \( | S \rangle \):

\[
| A \rangle | S \rangle = \Omega(| A \rangle | 0 \rangle),
\]

with \( | S \rangle \) initially in state \( | 0 \rangle \). Let \( \Delta \) be an operator which computes a superposition of the makespan of all schedules:

\[
| A \rangle | S \rangle | C_{max} \rangle = \Delta(| A \rangle | S \rangle | 0 \rangle) = \Delta \Omega(| A \rangle | 0 \rangle | 0 \rangle).
\]

Figure outlines our procedure to produce the vector \( | C_{max} \rangle \), a superposition of the makespans of all schedules. We now turn our attention to quantum circuits able to carry out the transformations discussed in this section and to the question how to obtain from the superposition of all makespans the optimal one.
Figure 1: Quantum circuit to prepare the makespan vector. First we create a superposition of all possible schedules $|\mathcal{A}^{(Nmq)}\rangle$ using a quantum circuit $\Omega$ having as input the $N$ job state vectors $|J_i\rangle$ and a register of $mq$ qubits in state $|0\rangle$. Then, we construct the vector $|S^{(mq)}\rangle$, a superposition of the running time on each machine for every schedule. The notation $|J_i^{(mq)}\rangle$ indicates that the vector $|J_i\rangle$ consists of $mq$ qubits. Similarly, $|\mathcal{A}^{(Nmq)}\rangle$ indicates that $|\mathcal{A}\rangle$ consists of $Nmq$ qubits.

First, we need to prepare job vector $|J_i\rangle$ in an equal superposition state which includes the processing times of job $J_i$ on all machines. The circuit in Figure 2 allows us to prepare job-state vectors. We use $m$ index qubits to control the job-machine information encoding. When each index qubit is set to $1/\sqrt{2}(|0\rangle + |1\rangle)$, the target qubits will be prepared in superposition of all possible job-machine states, $e_{ij}$, $1 \leq i \leq n$, $1 \leq j \leq m$. Figure 3 shows the circuit used to prepare the job vector $|J_1\rangle$ for our example.

Example: Table 3 summarizes the processing time of 8 jobs on 4 machines, when $0 < p_{ij} < 2^4 = 16$. Thus $n = 3$, $m = 2$, and $q = 4$. The running time of $J_1$ on machines $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ and $\mathcal{M}_4$ are respectively, 1, 3, 7, and 15 units of time.

| Job/|Machine | $\mathcal{M}_1$ | $\mathcal{M}_2$ | $\mathcal{M}_3$ | $\mathcal{M}_4$ |
|-----|---------|---------------|---------------|---------------|---------------|
| $J_1$ | 1 | 3 | 7 | 15 |
| $J_2$ | 2 | 1 | 9 | 3 |
| $J_3$ | 6 | 2 | 5 | 8 |
| $J_4$ | 11 | 13 | 7 | 4 |
| $J_5$ | 15 | 12 | 3 | 10 |
| $J_6$ | 10 | 7 | 8 | 14 |
| $J_7$ | 5 | 2 | 3 | 9 |
| $J_8$ | 1 | 10 | 11 | 13 |

The four vectors used to encode the processing time of job $J_1$ on machines $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ and $\mathcal{M}_4$ are respectively:

$|e_1^1\rangle = |000001\rangle$, $|e_2^1\rangle = |010011\rangle$, $|e_3^1\rangle = |100111\rangle$, $|e_4^1\rangle = |111111\rangle$.

For $J_2$ the basis vectors are $|000010\rangle$, $|010001\rangle$, $|101001\rangle$, $|110011\rangle$, and so on.
Figure 2: Quantum circuit to prepare the job state vectors. In this case $|x\rangle$ is a set of two control qubits. Each control qubit is set to $1/\sqrt{2}(|0\rangle + |1\rangle)$. The target qubits will be prepared in a superposition of all possible job-machine states.

\[ |x\rangle \]

\[ |\Omega\rangle \]

\[ p_1 \]

\[ p_2 \]

\[ p_3 \]

\[ p_4 \]

\[ |J\rangle \]

Figure 3: Quantum circuit to prepare $|J_1\rangle$. The execution times of $J_1$ are 1, 3, 7, and 15 units of time respectively. $|x\rangle$ are control qubits which control the selection of the four job-machine states, $e_{j1}^i$, $1 \leq j \leq 4$. The first two qubits of $|J_1\rangle$ record the machine index, and the remaining four qubits record the time if $J_1$ is assigned to the corresponding machine.

The job state vectors in equal superposition of basis states are:

\[ |J_1\rangle = \frac{1}{2} (|000001\rangle + |010011\rangle + |101110\rangle + |111111\rangle), \]

\[ |J_2\rangle = \frac{1}{2} (|000010\rangle + |010001\rangle + |101001\rangle + |110011\rangle), \]

\[ |J_3\rangle = \frac{1}{2} (|000110\rangle + |010010\rangle + |100101\rangle + |111000\rangle), \]

\[ |J_4\rangle = \frac{1}{2} (|001011\rangle + |011101\rangle + |100111\rangle + |110100\rangle), \]

\[ |J_5\rangle = \frac{1}{2} (|001111\rangle + |011100\rangle + |100011\rangle + |111010\rangle), \]

\[ |J_6\rangle = \frac{1}{2} (|001010\rangle + |010111\rangle + |101000\rangle + |111110\rangle), \]
\[ |J_7\rangle = \frac{1}{2} (|000101\rangle + |01010\rangle + |10011\rangle + |111001\rangle), \]
\[ |J_8\rangle = \frac{1}{2} (|000001\rangle + |011010\rangle + |101011\rangle + |111101\rangle). \]

We give two examples of schedules. Schedule \(A_1\) is:

\[ [J_1 \mapsto M_1, \ J_2 \mapsto M_2, \ J_3 \mapsto M_1, \ J_4 \mapsto M_4, \ J_5 \mapsto M_3, \ J_6 \mapsto M_2, \ J_7 \mapsto M_3, \ J_8 \mapsto M_1] \]

This schedule corresponds to the following job states:

\[ |J_1\rangle = |00\rangle |00\rangle |01\rangle |01\rangle |01\rangle |11\rangle |11\rangle |00\rangle \]
\[ |J_2\rangle = |01\rangle |01\rangle |00\rangle |01\rangle |11\rangle |11\rangle |00\rangle |00\rangle \]
\[ |J_3\rangle = |01\rangle |01\rangle |00\rangle |01\rangle |11\rangle |11\rangle |00\rangle |00\rangle \]
\[ |J_4\rangle = |11\rangle |11\rangle |10\rangle |10\rangle |11\rangle |11\rangle |00\rangle |00\rangle \]
\[ |J_5\rangle = |10\rangle |10\rangle |11\rangle |11\rangle |11\rangle |11\rangle |00\rangle |00\rangle \]
\[ |J_6\rangle = |01\rangle |01\rangle |00\rangle |01\rangle |11\rangle |11\rangle |00\rangle |00\rangle \]
\[ |J_7\rangle = |10\rangle |10\rangle |11\rangle |11\rangle |11\rangle |11\rangle |00\rangle |00\rangle \]
\[ |J_8\rangle = |01\rangle |01\rangle |00\rangle |01\rangle |11\rangle |11\rangle |00\rangle |00\rangle \]

The schedule vector is:

\[ |A_1\rangle = |00\rangle |00\rangle |01\rangle |01\rangle |01\rangle |11\rangle |11\rangle |00\rangle \]

The makespan of this schedule is equal to the largest completion time of all machines:

\[ C_{A_1}(M_1) = 1 + 6 + 1 = 8, \quad C_{A_1}(M_2) = 1 + 7 = 8, \quad C_{A_1}(M_3) = 3 + 3 = 6, \quad C_{A_1}(M_4) = 4. \]

Thus,

\[ C_{A_1}^{\text{max}} = 8. \]

Schedule \(A_2\) is:

\[ [J_1 \mapsto M_2, \ J_2 \mapsto M_1, \ J_3 \mapsto M_3, \ J_4 \mapsto M_4, \ J_5 \mapsto M_3, \ J_6 \mapsto M_2, \ J_7 \mapsto M_1, \ J_8 \mapsto M_2] \]

This schedule corresponds to the following job states:

\[ |J_1\rangle = |01\rangle |01\rangle |00\rangle |01\rangle |10\rangle |11\rangle |10\rangle |00\rangle \]
\[ |J_2\rangle = |00\rangle |00\rangle |01\rangle |01\rangle |10\rangle |11\rangle |10\rangle |00\rangle \]
\[ |J_3\rangle = |10\rangle |10\rangle |11\rangle |11\rangle |10\rangle |11\rangle |10\rangle |00\rangle \]
\[ |J_4\rangle = |10\rangle |10\rangle |11\rangle |11\rangle |10\rangle |11\rangle |10\rangle |00\rangle \]
\[ |J_5\rangle = |01\rangle |01\rangle |00\rangle |01\rangle |10\rangle |11\rangle |10\rangle |00\rangle \]
\[ |J_6\rangle = |10\rangle |10\rangle |11\rangle |11\rangle |10\rangle |11\rangle |10\rangle |00\rangle \]
\[ |J_7\rangle = |10\rangle |10\rangle |11\rangle |11\rangle |10\rangle |11\rangle |10\rangle |00\rangle \]
\[ |J_8\rangle = |01\rangle |01\rangle |00\rangle |01\rangle |10\rangle |11\rangle |10\rangle |00\rangle \]

The schedule vector is:

\[ |A_2\rangle = |01\rangle |01\rangle |00\rangle |01\rangle |10\rangle |11\rangle |10\rangle |00\rangle \]

The makespan of this schedule is equal to the largest completion time of all machines:

\[ C_{A_2}(M_1) = 2 + 5 = 7, \quad C_{A_2}(M_2) = 3 + 7 + 1 = 11, \quad C_{A_2}(M_3) = 5 + 3 = 8, \quad C_{A_2}(M_4) = 4. \]

Thus,

\[ C_{A_2}^{\text{max}} = 11. \]
Figure 4: (a) The measurement of a single qubit in state \( |\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle \) using a one-qubit measurement gate. The instrument displays the result 0 with probability \( p_0 = |\alpha_0|^2 \) and 1 with probability \( p_1 = |\alpha_1|^2 \). (b) We use a one-qubit measurement gate to measure only the first qubit of a register of \( n \) qubits in state \( |\psi^{(n)}\rangle = \alpha_0 |0\rangle |\psi^{(n-1)}_0\rangle + \alpha_1 |1\rangle |\psi^{(n-1)}_1\rangle \). If the result displayed by the instrument is 0 (when the single qubit measured is projected on \( |0\rangle \)) then the remaining \( n - 1 \) qubits are in state \( |\psi^{(n-1)}_0\rangle \); when the result displayed by the instrument is 1 (when the single qubit measured is projected on \( |1\rangle \)) then the remaining \( n - 1 \) qubits are in state \( |\psi^{(n-1)}_1\rangle \).

4 Quantum Measurements

Quantum measurements are subject to Born’s rule. This fundamental postulate\(^2\) of quantum mechanics is typically formulated as:

Consider a system in state \( |\psi\rangle \) and an observable \( \mathcal{A} \) with eigenvectors \( \{ |a_i\rangle \} \) and eigenvalues \( \{ a_i \} \). Then the probability for a measurement of the observable \( \mathcal{A} \) to yield a particular value \( a_i \) is:

\[
p(a_i) = | \langle a_i | \psi \rangle |^2 .
\]

Figure 4 illustrates the application of the Born rule for the measurement of the state of qubits. In Figure 4(b) we illustrate the generalized Born rule; we use a one-qubit measurement gate to measure only the first qubit of a register of \( n \) qubits in state \( |\psi^{(n)}\rangle = \alpha_0 |0\rangle |\psi^{(n-1)}_0\rangle + \alpha_1 |1\rangle |\psi^{(n-1)}_1\rangle \). Based upon the result of the measurement we decide that the remaining \( (n - 1) \) qubits are either in state \( |\psi^{(n-1)}_0\rangle \) or in \( |\psi^{(n-1)}_1\rangle \). We can also apply the generalized Born rule to measure first \( k \) qubits of an \( n \)-qubit register originally in state

\[
|\psi^{(n)}\rangle = \sum_{i=0}^{2^k - 1} a_i |i\rangle \otimes |\psi^{(n-k)}_i\rangle .
\]

When the result of the measurement performed on the first \( k \) qubits is \( x = r \) then we know that the remaining \( (n - k) \) qubits are in state \( |\psi^{(n-k)}_r\rangle \).

We now apply these ideas to a scheduling problem. Although we do not use the machine index qubits in following calculations, we keep in mind that the index qubits are entangled with the target or working qubits which give the execution time on that particular machine. If we measure the

\(^2\)Even though no violation of the Born’s rule has been observed experimentally it would be desirable to replace this postulate by a derivation based upon a set of fundamental assumptions. A. M. Gleason has presented a mathematical motivation for the form of Born’s rule [8] and more recent W. H. Zurek used a mechanism termed environment-assisted invariance, or envariance, for a novel derivation of the Born’s rule [31].
working qubits, the index qubits will be projected as well. If we have the means to identify the
optimal schedule and measure its working qubits, then the index qubits give the choice of machine
for each job, for the optimal schedule.

For example, consider the case the schedule vector is in an equal superposition of two schedules,
\( \mathcal{A}_1 \) and \( \mathcal{A}_2 \):

\[
| \mathcal{A} \rangle = \frac{1}{\sqrt{2}} (| \mathcal{A}_1 \rangle + | \mathcal{A}_2 \rangle)
\]

\[
= \frac{1}{\sqrt{2}} (| 000001 \rangle \otimes | 010011 \rangle \otimes | 001100 \rangle \otimes | 101000 \rangle
\]

\[
+ | 010011 \rangle \otimes | 000100 \rangle \otimes | 100101 \rangle \otimes | 111000 \rangle \otimes | 011000 \rangle)
\]

When we measure a subset of qubits, e.g. the the rightmost four (the ones which are not
underlined) and obtain the result

\[
| 0001 \rangle \otimes | 0001 \rangle \otimes | 0110 \rangle \otimes | 0100 \rangle \otimes | 0011 \rangle \otimes | 1111 \rangle \otimes | 0011 \rangle \otimes | 0001 \rangle
\]

we know the state of the remaining index qubits, which indicate the particular mapping of jobs to
machines given by schedule \( \mathcal{A}_1 \):

\( J_1 \mapsto M_1, \ J_2 \mapsto M_2, \ J_3 \mapsto M_1, \ J_4 \mapsto M_4, \ J_5 \mapsto M_3, \ J_6 \mapsto M_2, \ J_7 \mapsto M_3, \ J_8 \mapsto M_1. \)

5 The Running Time of Machine \( M_j \) under Schedule \( \mathcal{A}_k \)

Computing the total running time of machine \( M_j \) under schedule \( \mathcal{A}_k \) requires summation of the
running times of all jobs assigned to \( M_j \). Now we are concerned with the construction of a quantum
adder similar to a classic adder, e.g. \( | 5 \rangle + | 6 \rangle = | 11 \rangle \). A quantum adder has:

- \( n \) inputs \( a_1, a_2, \ldots, a_n \), each one is a register of \( q \) qubits,
- one carry in \( c \), and
- one output \( S = \sum_{i=1}^{n} a_i + c \), a register of \( q + n \) qubits.

The truth table of a one-qubit quantum adder is presented in Table 1 and the corresponding
quantum circuit is illustrated by Figure 5.

Table 1: The truth table of a one-qubit quantum adder with two inputs \( | x_i \rangle \) and \( | y_i \rangle \) and a carry
in, \( | c_i \rangle \).

| \( x_i \) | \( y_i \) | \( c_i \) | \( x_i + y_i \) |
|-------|-------|-------|-----------|
| 0     | 0     | 0     | 00        |
| 0     | 0     | 1     | 01        |
| 1     | 0     | 0     | 01        |
| 1     | 0     | 1     | 10        |
| 0     | 1     | 0     | 01        |
| 0     | 1     | 1     | 10        |
| 1     | 1     | 0     | 10        |
| 1     | 1     | 1     | 11        |
Figure 5: One-qubit quantum adder with carry in, $| c_i \rangle$, and two inputs $| x_i \rangle$ and $| y_i \rangle$. The outputs are entangled; once we measure $| x_i + y_i \rangle$, the inputs $x_i$ and $y_i$ and the carry in $| c_i \rangle$ are affected.

For example, when

$$| x_i \rangle = \frac{1}{\sqrt{2}} (| 0 \rangle + | 1 \rangle), \quad | y_i \rangle = | 1 \rangle, \quad | c_i \rangle = | 0 \rangle$$

the output is:

$$\frac{1}{\sqrt{2}} (| 01001 \rangle + | 11010 \rangle).$$

If we only measure the last two qubits, the summation qubits, the output of the circuit is:

$$\frac{1}{\sqrt{2}} | x_i + y_i \rangle = \frac{1}{\sqrt{2}} (| 01 \rangle + | 10 \rangle).$$

Once we measure $| x_i + y_i \rangle$ and obtain the result $1/\sqrt{2} | 10 \rangle$, we are certain the input $| x_i \rangle = 1/\sqrt{2} | 1 \rangle$, as the result of the addition and the original inputs are entangled.

One may argue that a quantum adder violates the no cloning theorem. If one can build a circuit performing the function

$$| x_i \rangle + | y_i \rangle = | z_i \rangle$$

when we set $| y_i \rangle = | 0 \rangle$, then $| z_i \rangle = | x_i \rangle$. The circuit in Figure 6 produces two entangled qubits rather than a replica of its input, $| \psi \rangle = a | 0 \rangle + b | 1 \rangle$.

Figure 6: Circuit whose output is $a | 00 \rangle + b | 11 \rangle$ rather than $(a | 0 \rangle + b | 1 \rangle) \otimes (a | 0 \rangle + b | 1 \rangle)$.

Similarly, the quantum circuit in Figure 6 does not violate the no cloning theorem. Unlike a classic adder when $x_i, y_i$, and $x_i + y_i$ can be used separately, a quantum adder produces $| x_i \rangle, | y_i \rangle$, and $| x_i + y_i \rangle$ which are entangled. After a measurement of one of them, the other two will be altered.

To obtain the total running time of machine $M_j$ for schedule $A_k$ we add the execution times, (in our examples those qubits of a job-machine vector which are not underlined) for all jobs assigned to $M_j$ and create a running time vector for machine $M_j$, $| S_j \rangle$. The running time vector for schedule $A_k$ is the superposition of the execution time on individual machines:
\[ |S^{A_k}⟩ = |S_1^{A_k}⟩ \otimes |S_2^{A_k}⟩ \otimes \cdots \otimes |S_m^{A_k}⟩. \]

For example, for machine \(M_1\) under schedule \(A_1\) (see Section 3):

\[ |S_1⟩ : |0001⟩ + |0000⟩ + |0110⟩ + |0000⟩ + |0000⟩ + |0000⟩ + |0000⟩ + |0001⟩, \]

or

\[ |S_1⟩ : |0001000⟩. \]

Figure 7: A circuit to compute the sum of the execution time of all jobs assigned to each machine.

\(|J_1⟩, |J_2⟩ \ldots |J_N⟩\) are the job vectors and \(|S_1⟩, |S_2⟩ \ldots |S_M⟩\) represent the execution time on machines \(M_1, M_2, \ldots M_M\) respectively.

We wish to construct a quantum circuit to add the execution time for all jobs assigned to any one machine \(M_j\). We use the index qubits as control qubits to control the summation of each index, or each machine; the index qubits are entangled with the target qubits which give the running time. In Figure 7 \(|S_1⟩, |S_2⟩ \ldots |S_M⟩\) represent the execution time on machines \(M_1, M_2, \ldots M_M\) respectively. When the index qubits are not “active”, the respective input for the summation circuit will be zero. For example, if the index of \(e_2^2 = |010001⟩\) is \(01\), rather than \(00\), then the index qubits of \(e_2^2\) for \(|S_1⟩\) are not active.

We need at most \(q\) qubits to express the execution time of any job on any machine. Thus the sum of the execution time of at most \(2^n\) jobs, will be at most equal to \(2^{n+q}\) and we need \(n+q\) qubits to express it.

In our further calculations we only use \(|S_j⟩\) and do not touch \(|J_i⟩\). For the example in Section 4 the system will be in the state:

\[ |A⟩ \otimes |S⟩ = 1/\sqrt{2}(|A_1 S_1⟩ + |A_2 S_2⟩) \]

or

\[ |A S⟩ = 1/\sqrt{2}(|000001⟩ \otimes |010001⟩ \otimes |000110⟩ \otimes |110100⟩ \otimes |100011⟩ \otimes |010111⟩ \otimes |010001⟩ \otimes |000010⟩ \otimes |000100⟩) \otimes |000001⟩ \otimes |000110⟩ \otimes |110100⟩ \otimes |100011⟩ \otimes |010111⟩ \otimes |010001⟩ \otimes |000010⟩ \otimes |000100⟩) \)
6 Determine the Makespan

The vector $|S_j\rangle$ provides the running time of machine $M_j$ under all schedules. The makespan of a schedule is equal to the maximum running time of any machine under that schedule. We want to construct a “Max” circuit, as shown in Figure 8, to get the makespan of each schedule. The quantum circuit computes the maximum over an input set, e.g., Max($|5\rangle, |7\rangle, |4\rangle) = |7\rangle$. The input to this circuit is a register of circuit of $n + q$ qubits. The output $|\text{Max}\rangle$, has also $n + q$ qubits.

![Figure 8: The quantum circuit to compute the makespan, the maximum running time among all machines](image)

We face now the same problem we discussed in the previous section. The output is entangled with the input, it is not simply $|S_1\rangle \otimes |S_2\rangle \otimes \cdots \otimes |S_M\rangle \otimes |\text{Max}\rangle$. This means that once we measure the register containing the vector $|\text{Max}\rangle$ we affect all the vectors $|S_1\rangle, |S_2\rangle, \ldots, |S_M\rangle$.

![Figure 9: A two-qubit quantum circuit for the Max function. The circuit has two 2-qubit input registers $|x\rangle, |y\rangle$, two ancilla qubits initially in state $|0\rangle$ and a two-qubit register for the Max value, initially set to $|00\rangle$.](image)

Figure 9 shows a Max quantum circuit for two 2-qubit inputs $|x\rangle$ and $|y\rangle$ whose table of truth is given in Table 2. $|a\rangle$ and $|b\rangle$ are ancilla qubits. To see how the output qubits are entangled let

$$
|x\rangle = 1/\sqrt{2}(|01\rangle + |11\rangle), \quad |y\rangle = 1/\sqrt{2}(|01\rangle + |10\rangle).
$$

Then,
Table 2: The truth table for Max quantum circuit with two 2-qubit input registers $|x\rangle$, $|y\rangle$.

|   |   | $|\text{Max}\rangle$ |   |   | $|\text{Max}\rangle$ |
|---|---|-------------------|---|---|-------------------|
| 00 | 00 | 00                | 00 | 10 | 10                |
| 01 | 00 | 01                | 01 | 10 | 10                |
| 10 | 00 | 10                | 10 | 10 | 10                |
| 11 | 00 | 11                | 11 | 10 | 11                |
| 00 | 01 | 01                | 00 | 11 | 11                |
| 01 | 01 | 01                | 01 | 11 | 11                |
| 10 | 01 | 10                | 10 | 11 | 11                |
| 11 | 01 | 11                | 11 | 11 | 11                |

$|\text{Max}\rangle = 1/2(|11\rangle + |11\rangle + |10\rangle + |01\rangle)$.

Now if we measure the register $|\text{Max}\rangle$ and we observe two qubits in state $|01\rangle$, we are sure that $x$ and $y$ are: $|x\rangle = |01\rangle$ and $|y\rangle = (|01\rangle$. Here the $|\text{Max}\rangle$ qubits are in a superposition of the makespans of all schedules. If we measure the qubits associated the makespan and determine that they represent the state to $|\mu\rangle$, than the qubits $|S_j\rangle$ which reveal the machines each task is allocated to, will be forced into the corresponding states of the schedule with a makespan equal to $\mu$.

For the example in Section 4, the system will be in the state:

$|\text{Max}\rangle = 1/2(|A\rangle \otimes |S\rangle \otimes |\text{Max}\rangle + |A_2\rangle \otimes |S_2\rangle \otimes |\text{Max}_2\rangle)$

$= 1/\sqrt{2}(|000001\rangle \otimes |010001\rangle \otimes |000110\rangle \otimes |110100\rangle \otimes |100011\rangle \otimes |010111\rangle \otimes |100011\rangle$

$\otimes |011001\rangle \otimes |000100\rangle \otimes |001000\rangle \otimes |000110\rangle \otimes |000010\rangle \otimes |000100\rangle$

$\otimes |010011\rangle \otimes |000010\rangle \otimes |100101\rangle \otimes |101001\rangle \otimes |110100\rangle \otimes |100011\rangle \otimes |011101\rangle \otimes |000101\rangle \otimes |000111\rangle \otimes |001011\rangle \otimes |000110\rangle \otimes |000010\rangle \otimes |000101\rangle \otimes |000101\rangle\rangle$

7 Find a Schedule with a Given Makespan

We use a generalized version of Grover’s search algorithm to find a schedule with a given makespan. The basic ideas of Grover’s quantum search algorithm are discussed next. We create an equal superposition of the indices of all elements. Initially, all possible outcomes are equally likely and have the same amplitude. An oracle examines an index/label, $x$, and decides if it matches the search argument or not. Sets $f(x) = 0$ if $x$ is not a solution and $f(x) = 1$ if a solution. The phase of the “solution” is rotated by $2\theta$ radians. At each iteration the mean value of the state is the same, but the amplitude of the “solution” is amplified.

Consider a search space $S_{\text{search}} = \{E_x\}$ consisting of $N = 2^n$ elements. Each element $E_x, 1 \leq x \leq 2^n$, is uniquely identified by a binary $n$-tuple $x$, called the index of the element. We assume that $M \leq N$ elements satisfy the requirements of a query and we wish to identify one of them.

The classic approach is to repeatedly select an element $E_j$, decide if the element is a solution to the query, and if so, terminate the search. If there is a single solution ($M = 1$) then a classical search algorithm requires $O(2^n)$ iterations.

The basic idea of Grover’s quantum search algorithm is illustrated in Figure 10. We apply a Walsh-Hadamard transform to create an equal superposition state:

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

Then we perform Grover’s iterations. The circuit for this algorithm is presented in Figure 11.
To abstract this process we consider a function $f(x)$ with $0 \leq x \leq 2^n - 1$ such that

$$f(x) = \begin{cases} 0 & \text{if } x \text{ is not a solution} \\ 1 & \text{if } x \text{ is a solution} \end{cases}.$$  

Consider an oracle, a black box accepting as input $n$ qubits representing the index $x$ of an element $E_x, 1 \leq x \leq 2^n$. The black box also has an oracle qubit, $|q\rangle$, initially set to $|0\rangle$ and reset to $|1\rangle$ when the oracle recognizes a solution to the search problem we pose. The black box performs the following transformation

$$|x\rangle |q\rangle \rightarrow |x\rangle |q \oplus f(x)\rangle.$$  

The oracle qubit can be initialized in the state

$$|q\rangle = (1/\sqrt{2})(|0\rangle - |1\rangle).$$

If an input $|x\rangle$ is not a solution to our search problem, then the oracle qubit is unchanged. On the other hand, a solution to the search problem transforms $|0\rangle$ into $|1\rangle$. Thus, the transformation performed by the black box is

$$|x\rangle (|0\rangle - |1\rangle)/\sqrt{2} \rightarrow \begin{cases} |x\rangle (|0\rangle - |1\rangle)/\sqrt{2} & \text{if } f(x) = 0 \implies E_x \text{ is not a solution} \\ -|x\rangle (|0\rangle - |1\rangle)/\sqrt{2} & \text{if } f(x) = 1 \implies E_x \text{ is a solution}. \end{cases}$$

This transformation can be rewritten as

$$|x\rangle (|0\rangle - |1\rangle)/\sqrt{2} \rightarrow (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle)/\sqrt{2}.$$
The state of the oracle qubit does not change and can be omitted from the description of the quantum search algorithm

\[ |x\rangle \rightarrow (-1)^{f(x)} |x\rangle. \]

Let \( U \) be the following transformation:

\[
\begin{align*}
U = 2 |0\rangle \langle 0| - I.
\end{align*}
\]

Then a conditional phase shift, Figure 12 applied to the system in state \( |\psi\rangle \) is:

\[
S_p = H^\otimes n U H^\otimes n = H^\otimes n (2 |0\rangle \langle 0| - I) H^\otimes n = 2 |0\rangle \langle 0| - I.
\]

A Grover’s iteration consists of \( O \), the transformation performed by the oracle followed by a conditional phase shift:

\[
G = S_p O = (2 |0\rangle \langle 0| - I) O
\]

Figure 13: Grover’s operator as a rotation operator.

Grover’s operator

\[
G = (2 |0\rangle \langle 0| - I) O
\]

is a rotation operator, Figure 13. If \( N \) is the number of elements in the database and \( M \) the number of solutions, let \( \alpha \) and \( \beta \) be:
\[ |\alpha\rangle = \frac{1}{\sqrt{N-M}} \sum_{x \text{ is not a solution}} |x\rangle \quad \text{and} \quad |\beta\rangle = \frac{1}{\sqrt{N-M}} \sum_{x \text{ is a solution}} |x\rangle \]

We define the orthonormal basis formed by \( \alpha \) and \( \beta \). Therefore:

\[ |\psi\rangle = \sqrt{\frac{N-M}{N}} |\alpha\rangle + \sqrt{\frac{M}{N}} |\beta\rangle \]

or

\[ |\psi\rangle = \cos(\theta/2) |\alpha\rangle + \sin(\theta/2) |\beta\rangle \]

with

\[ \cos(\theta/2) = \sqrt{\frac{N-M}{N}}. \]

After \( k \) Grover’s iterations the state of the system is:

\[ G^k |\psi\rangle = \cos((2k+1)\theta/2) |\alpha\rangle + \sin((2k+1)\theta/2) |\beta\rangle. \]

Based upon this geometric interpretation of Grover’s iteration let us now attempt to answer the question on how many iterations are needed when \( M \leq N/2 \). At each iteration the state is rotated with an angle \( 2\theta \). Thus the number of iterations is:

\[ R \leq \left\lceil \frac{\pi}{2\theta} \right\rceil \]

with

\[ \frac{\theta}{2} \geq \sin(\theta/2) = \sqrt{\frac{M}{N}}. \]

Thus

\[ R \leq \left\lceil \frac{\pi}{4\sqrt{\frac{N}{M}}} \right\rceil \]

or

\[ R = O\left(\sqrt{\frac{N}{M}}\right). \]

More recently Grover’s algorithm was generalized to the case in which the initial amplitudes are either real or complex and follow any arbitrary distribution \([1, 2]\).

Now let us return to our scheduling problem and recall that:

- We prepare each job vector \( |J_i\rangle \) in a superposition state which includes the running times on all machines. The first \( m \) qubits of a job vector are used for index of the machine and remaining \( q \) qubits are used for the running time of that job on the machine.

- We add the execution time of all jobs according to their machine indexes, which gives us the running time of each machine \( |S_j\rangle \). The qubits \( |S\rangle = |S_1\rangle \otimes |S_1\rangle \cdots \otimes |S_m\rangle \) are entangled with the qubits of the job vectors \( |J\rangle \), which means if we project or output one special \( |S\rangle \), the job vectors will change into a special set according to the \( |S\rangle \). The index qubits in job vectors will give us the details of the special schedule.

- We get the maximum running time among all machines using the Max quantum circuit. The \( |\text{Max}\rangle \) qubits are entangled with \( |S\rangle \) and \( |J\rangle \) qubits. If we project (or measure) one special \( |\text{Max}\rangle \) (e.g., the best one), the \( |S\rangle \) and \( |J\rangle \) set will change into one special set according to the projection of the \( |\text{Max}\rangle \) qubits (or more than one if possible).
The $|\text{Max}\rangle$ qubits we obtain represent a superposition of the makespans, $D_k$, of all possible schedules. If we measure the makespan equal to $\mu$, the job vectors give us the details of the schedule, as they are entangled with the qubits for the makespan!

Let us now examine $|\text{Max}\rangle$, a superposition of the makespans of all schedules. We use a generalized Grover search algorithm to find the schedule with the makespan $D_k = \mu$. If we find such a makespan, each job vector, will allow us to determine the schedule.

Our search algorithm is essentially a Grover’s algorithm, but without the initialization step. The search process starts with arbitrary initial amplitudes and proceeds as follows:

- Use the output ($|\text{Max}\rangle$) of our scheduling. The state $|\text{Max}\rangle$ is not an equal superposition of the component vectors.
- Repeat the following steps $T$ times:
  1. Rotate the result states by $\pi$ radians.
  2. Apply the “inversion about average” operator to rotate all states by $\pi$ radians around the average amplitude of all states.
- Measure the resulting state.

Given an arbitrary initial distribution of $r$ marked and $N - r$ unmarked states, with known average $\overline{k}(0)$ and $\overline{l}(0)$ respectively, the optimal measurement requires $T$ iterations \[ \text{with } T \text{ given by:} \]

\[
T = \frac{(j + 1/2)\pi - \arctan\left(\frac{\sqrt{Q/r}}{2r/Q}\right)}{\arccos(1 - 2r/Q)}, \quad j = 0, 1, 2, \ldots
\]

In this expression $Q$ is the size of the searching space, in our case equal to $2^{(n+q)}$ as $|\text{Max}\rangle$ has $n + q$ qubits. $r$ is the number of the solutions; we assume that there is only one solution, thus $r = 1$. $\overline{k}(0)$ is the average amplitude of solution states and $\overline{l}(0)$ is the average amplitude of states which are not solutions.

When $r \ll Q$, the above equation yields:

\[
T = -\frac{\overline{k}(0)}{2\overline{l}(0)} + \frac{\pi}{4} \sqrt{Q/r} - \frac{\pi}{24} \sqrt{r/Q} + O(r/Q).
\]

The total number of schedules is $M^N$ as we have $N = 2^n$ jobs running on $M = 2^m$ machines, so:

\[
\overline{k}(0) \sim \sqrt{\frac{1}{M^N}}
\]

The number of different schedules, $M^N$, is much larger than $Q = 2^{(m+q)}$, the total number of states. We can assume they are uniformly distributed with the average:

\[
\overline{l}(0) \sim \sqrt{\frac{1}{2^{(m+q)}}}
\]

Then,

\[
\frac{\overline{k}(0)}{2\overline{l}(0)} \sim \sqrt{\frac{2^{(m+q)}}{M^N}} \to 0 \quad \Rightarrow \quad T \sim \frac{\pi}{4} \sqrt{Q}
\]

As the register holding $|\text{Max}\rangle$ has $n + q$ qubits, we need $T \sim \mathcal{O}(\sqrt{2^{n+q}})$ Grover iterations and succeed with probability $\mathcal{O}(1)$. 17
8 Summary

The algorithm discussed in this paper makes extensive use of a generalized version of Grover’s search algorithm to solve the following problem: given a set of $N = 2^n$ jobs $J_i$, $1 \leq i \leq 2^n$, a set of $M = 2^m$ machines, $M_j$, $1 \leq j \leq 2^m$ and the execution time of all the jobs on all machines, $1 \leq p_{ij} \leq 2^q$, does a schedule $A$ with the makespan $C_{max}^A = \mu$ exist, and if it does, what is the schedule $A$. Given a range of makespans $\mu_{low} \leq \mu \leq \mu_{high}$ we can use the algorithm repeatedly to find a schedule with a makespan in that range.

The algorithm constructs a superposition of all schedules and a superposition of their makespans and then amplifies the one that corresponds to the solution. We perform $O(\sqrt{2^n+q})$ Grover iterations. The time complexity of the quantum scheduling algorithm for an $R||C_{max}$ problem is $O(\sqrt{2^n+q})$ and the corresponding complexity of a classical algorithm is $O(2^m2^n)$.

9 Acknowledgments

The research reported in this paper was partially supported by National Science Foundation grant CCF 0523603.

References

[1] D. Biron, O. Biham, E. Biham, M. Grassl, D. A. Lidar. *Generalized Grover Search Algorithm for Arbitrary Initial Amplitude Distribution*. Preprint, [http://arxiv.org/archive/quant-ph/9801066](http://arxiv.org/archive/quant-ph/9801066), 1998.

[2] E. Biham, O. Biham, D. Biron, M. Grassl, D. A. Lidar, and D. Shapira. *Analysis of Generalized Grover Search Algorithms for Using Recursion Equations*. Preprint, [http://arxiv.org/archive/quant-ph/0010077](http://arxiv.org/archive/quant-ph/0010077), 2000.

[3] A. Berthiaume and G. Brassard. *Oracle Quantum Computing*. Proc. Workshop on Physics of Computation, IEEE Pres, Los Alamitos Ca, 195–199, 1992.

[4] R. Cleve, A. Ekert, L. Henderson, C. Macchiavello, and M. Mosca. *On Quantum Algorithms*. Preprint, [http://arxiv.org/archive/quant-ph/9903061](http://arxiv.org/archive/quant-ph/9903061), v1, 1999.

[5] A. Ekert, R. Josza. *Quantum Algorithms: Entanglement Enhanced Information Processing*. Preprint, [http://arxiv.org/archive/quant-ph/9803072](http://arxiv.org/archive/quant-ph/9803072), v1, November 2000, also in Phyl., Trans., Roy., Soc., London, 1997.

[6] A. Ekert, P. Hayden, and H. Inamori. *Basic Concepts in Quantum Computing*. Preprint, [http://arxiv.org/archive/quant-ph/0011013](http://arxiv.org/archive/quant-ph/0011013), v1, November 2000.

[7] R. P. Feynman. *Simulating Physics with Computers*. Int. J. Theoret. Phys. 21:467–488, 1982.

[8] R. P. Feynman. *Quantum Mechanical Computers*. Found. Phys. 16:507–531, 1986.

[9] A. M. Gleason. *Measures on a Close Subspace of a Hilbert Space*. J. Math. Mec., 6, 885, 1957.

[10] L. K. Grover. *A Fast Quantum Algorithm for Database Search*. Proc. ACM Symp. on Theory of Computing, ACM Press, 212–219, 1996.

[11] L. K. Grover. *Quantum Mechanics Helps in Searching for a Needle in a Haystack*. Phys. Rev. Lett. 78, 325–328, 1997.

[12] L. K. Grover. *A Framework for Fast Quantum Mechanical Algorithms*. Proc. Symp. on Theory of Computing, ACM Press, 53–62, 1998.
[13] R. Josza. *Entanglement and Quantum Computation*. Preprint, http://arxiv.org/archive/quant-ph/9707034 v1, 1997, also Geometric Issues in the Foundations of Science. S. Hugget, L. Mason, K.P. Tod, S.T. Tsou, and N. M. J. Woodhouse, Editors. Oxford University Press, 1997.

[14] R. Josza. *Quantum Algorithms and the Fourier Transform*. Preprint, http://arxiv.org/archive/quant-ph/9707033 v1, 1997, also Proc., Roy., Soc., London, 1997.

[15] R. Josza. *Searching in Grover’s Algorithm*. Preprint, http://arxiv.org/archive/quant-ph/9901021 v1, 1999.

[16] R. Josza. *Quantum Factoring, Discrete Logarithms, and the Hidden Subgroup Problem*. Preprint, http://arxiv.org/archive/quant-ph/0012084 v1, 2000.

[17] R. Josza. *Illustrating the Concept of Quantum Information*. Preprint, http://arxiv.org/archive/quant-ph/0305114 v1, 2003.

[18] A. Yu. Kitaev *Quantum Measurements and the Abelian Stabilizer Problem*. Preprint, http://arxiv.org/archive/quant-ph/9511026 v1, 1995.

[19] E. Knill, R. Laflame, H. Barnum, D. Dalvit, J. Dziarmaga, J. Gubernatis, L. Gurvis, G. Ortiz, L. Viola, and W. H. Zurek. *Quantum Information Processing*. Kluwer Encyclopedia of Mathematics, Suplement III, 2002.

[20] D. C. Marinescu and G. M. Marinescu. *Approaching Quantum Computing*. Prentice Hall, Upper Saddle River, NJ, ISBN 0-13-145224-X, 2004.

[21] M. Mosca and A. Ekert. *The Hidden Subgroup Problem and Eigenvalue Estimation on a Quantum Computer*. Preprint, http://arxiv.org/archive/quant-ph/9903071 v1, May 1999.

[22] M.A. Nielsen and I. L. Chuang. *Quantum Computing and Quantum Information*. Cambridge University Press, ISBN 0-521-63245-8, 2000.

[23] C. M. Papadimitriou. Computational Complexity. Addison-Wesley, Reading, Mass., ISBN 0-201-53082-1, 1994.

[24] P. W. Shor. *Algorithms for Quantum Computation: Discrete Log and Factoring*. Proc. 35 Annual Symp. on Foundations of Computer Science, pages 124–134, IEEE Press, Piscataway, New Jersey, 1994.

[25] P. W. Shor. *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. SIAM J. Computing 26:1484–1509, 1997.

[26] P. W. Shor. *Introduction to Quantum Algorithms*. Preprint, http://www.arXiv/quant-ph/0005003 July, 2001.

[27] P. W. Shor. *Why Haven’t More Quantum Algorithms Been Found*. Journal of the ACM, 50(1): 87–90, 2003.

[28] D. R. Simon. *On the Power of Quantum Computation*. SIAM J. Computing 26:1474–1483, 1997.

[29] A. Steane. *Quantum Computing*. Reports Prog. Phys. 61, 117, 1998. Preprint, http://arxiv.org/archive/quant-ph/97080222 v2, September 1997.

[30] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. S. Chuang. *Experimental Realization of Shor’s Quantum Factoring Algorithm Using Nuclear Magnetic Resonance*. Nature, 414, 883:887, 2001. Also, Preprint, http://arxiv.org/archive/quant-ph/0112176 v1, December, 2001.
[31] W. H. Zurek. *Probabilities from Envariance.* Preprint, http://arxiv.org/archive/quant-ph/0405161, 2004.