UJMM ONE + TWO

Undergraduate Journal of Mathematical Modeling: One + Two

Volume 4 | 2012 Spring

Issue 2 | Article 2

The Subset Sum Problem: Reducing Time Complexity of NP-Completeness with Quantum Search

Bo Moon
University of South Florida

Advisors:
Manoug Manougian, Mathematics and Statistics
Jing Wang, Computer Science & Engineering

Problem Suggested By: Jing Wang

Abstract. The Subset Sum Problem is a member of the NP-complete class, so no known polynomial time algorithm exists for it. Although there are polynomial time approximations and heuristics, these are not always acceptable, yet exact-solution algorithms are unfeasible for large input. Quantum computation offers new insights for not only the Subset Sum Problem but also the entire NP-complete class; most notably, Grover's quantum algorithm for an unstructured database search can be tailored to identify solutions to problems within mathematics and computer science. This paper discusses the physical and conceptual feasibility of quantum computation and demonstrates the utility of quantum search by analyzing the time complexities of the classical dynamic programming algorithm and Grover's algorithm in solving the Subset Sum Problem, evincing the implications this has on the NP-complete class in general.

Keywords. NP Completeness, Quantum Search, Subset Sum Problem

TABLE OF CONTENTS

PROBLEM STATEMENT

The Subset Sum Problem is a member of the NP-complete class of computational problems, having no known polynomial time algorithm. The purpose of this paper is to compare the time complexities of a quantum search algorithm and a classical dynamic programming algorithm as solutions to this problem.

MOTIVATION

In computational complexity theory, problems within the NP-complete class have no known algorithms that run in polynomial time. The study of NP-completeness is significant, as computer science problems lurk in many guises across a variety of disciplines, from chemical informatics to networking. As such, identifying a problem as NP-complete will conserve both time and effort for the computer scientist, who can avoid a fruitless pursuit for an efficient algorithm by knowing beforehand that there are currently none in existence. Of course, NP-complete problems can still be solved, but either the input data must be restricted to reasonably small sizes to accommodate superpolynomial time algorithms or accuracy must be compromised in implementing faster approximation algorithms, neither of which are amenable conditions.

Although most algorithms for NP-complete problems are just short of a brute-force search, the immense computational power of a quantum computer may give impetus to more efficient solutions. A quantum search algorithm—namely Grover's algorithm—provides a framework for finding a solution in a finite search space defined in terms of the problem, running faster than a classical algorithm but admittedly still exponential. As such, this paper will demonstrate the utility of quantum search in solving the NP-complete class by applying Grover's algorithm to a specific instance of NP-completeness—the Subset Sum Problem. As it shall be

http://scholarcommons.usf.edu/ujmm/vol4/iss2/2
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.2

soon revealed, a classical algorithm can do no better than a complete search while a quantum search runs drastically faster by exploiting the principles of superposition and quantum parallelism on an unstructured database.

MATHEMATICAL DESCRIPTION AND SOLUTION APPROACH

A. THE SUBSET SUM PROBLEM AND DYNAMIC PROGRAMMING

The Subset Sum Problem is as follows: Given a set S of n positive integers and a positive target integer T, determine whether there exists a subset of S whose elements sum to T (Neapolitan and Naimipour). This problem has been shown to be NP-complete by reduction to the satisfiability problem, so no known polynomial time algorithm exists (Dasgupta, Papadimitriou and Vazirani). One possible algorithm is to generate all $2^n - 1$ possible nonempty subsets of S through a depth-first search and to sum the elements in each set in $O(n)$ time, terminating only when T is reached. This yields a complete search algorithm with a runtime of $O(n2^n)$. A slightly faster algorithm in pseudopolynomial time can be achieved with dynamic programming.

In order to apply dynamic programming, the Subset Sum Problem must exhibit optimal substructure and overlapping subproblems. Optimal substructure appears when the solution to a problem relies on the solutions to smaller cases. In the Subset Sum Problem, suppose that one element x_j of the solution subset is known. The original problem is now reduced to finding a subset of $n - 1$ elements that adds up to $T - x_j$, so this subproblem consists of fewer elements and a smaller sum. Thus, an algorithm for the Subset Sum Problem can utilize optimal substructure by iterating over all x_i to create the subproblems, iterating over x_i recursively on

those subproblems until a base case is reached, and then conflating the solutions in order to solve the original problem, thereby reducing the number of time-consuming operations done. Overlapping subproblems occur when multiple subproblems consisting of identical parameters are solved. Consider two subproblems with sets S_1 and S_2 and corresponding target sums T_1 and T_2 such that $S_1 \neq S_2$ but $T_1 = T_2$. If it is possible to solve either case, then solving the other is unnecessary, as the succeeding case that gave rise to these subproblems requires only the fact whether or not it is possible to reach the sum regardless of the set used. As a result, overlapping subproblems can be avoided by recording solutions to subproblems in a table as they are solved and referring to it to determine if a particular recursive call is unique or has already been visited, so fewer cases are solved. Since the Subset Sum Problem possesses both optimal substructure and overlapping subproblems, a dynamic programming algorithm can be applied.

To exploit optimal substructure, a recursive relationship must be identified such that solutions to problems of smaller dimensionality are somehow combined. Let the function $M(i, j)$ return a Boolean value of true if it is possible to add to i using the jth element in the set and false otherwise. Given a target sum i and an element x_j, the only valid operations are to either use the element, resulting in a subproblem with target sum $i - x_j$ and element x_{j-1}, or skip the element, creating another subproblem with target sum i and element x_{j-1}. Using a logical OR operator on these two cases in functional form will determine whether it is possible to solve at least one of them and ultimately the previous recursive call as well. The function returns 1 when $i = 0$, indicating that the target sum has been successfully reached, and returns 0 when either i is negative or $j = 0$, which means either the subset sum has surpassed the target or the subset is out of elements respectively. The subproblems and the base cases can be represented as

http://scholarcommons.usf.edu/ujmm/vol4/iss2/2
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.2

$$M(i,j) = \begin{cases} 1 & if\ i = 0 \\ 0 & if\ i < 0\ or\ j = 0 \\ M(i - x_j, j-1) \vee M(i, j-1) & otherwise \end{cases}$$

so that $M(T, n)$ will solve the original problem statement.

To avoid overlapping subproblems, the algorithm uses a matrix where the element at the ith row and jth column conveniently holds the value $M(i, j)$. In implementing the function, one would refer to the matrix prior to evaluating each recursive call in order to check whether the particular subproblem has already been solved. If so, the element is retrieved, otherwise the matrix entry is filled in as soon as $M(i, j)$ is resolved so that future subproblems can refer to it.

B. TIME COMPLEXITY OF THE DYNAMIC PROGRAMMING ALGORITHM

Since this dynamic programming algorithm solves each subproblem exactly once and the space of all subproblems can be represented by an $i \times j$ matrix, the time complexity of this algorithm is equivalent to the total number of elements in the matrix. With T rows and n columns, there are Tn subproblems. Therefore, the time and space complexity of this dynamic programming algorithm is $O(Tn)$. For a more thorough evaluation of time complexity, let W be the number of distinct sums that must be created with the given set so that the following cases can explain its pseudopolynomial time behavior.

In the worst case scenario, T is greater than or equal to the sum of all elements in the set. As a result, the algorithm must visit every possible sum in order to determine if T can be reached at all. By elementary number theory, there are $2^n - 1$ distinct nonempty subsets, so W is bounded above as such:

$$W \leq 2^n - 1 = O(2^n).$$

Therefore, the original time complexity $O(Wn)$ (where $W = T$) can be rewritten as $O(n2^n)$.

Consider the alternate case when T is less than the sum of all elements in the set. The algorithm does not consider every possible subset sum since some would clearly be greater than T. Consequently, the algorithm visits only the unique sums $1, 2, \dots, T$ under the worst conditions for this particular case, so $W = T$. For example, consider the set $S = \{1, 2, 4, 8, 16\}$ with $T = 17$. There are $2^5 - 1 = 35$ possible subset sums, but only the sums $1, 2, \dots, 17$ will be considered because any greater sum is unnecessary, so $W = T = 17$. Therefore, these conditions preserve the $O(Tn)$ complexity.

This dynamic programming algorithm is considered to be pseudopolynomial because it behaves as a polynomial time algorithm for large elements in S and relatively small T, but it is not actually polynomial time as previously shown. However, it is reasonable to conclude that its runtime is $O(n2^n)$ because this represents the worst-case conditions according to order of growth analysis, and one cannot ensure that T is indeed bounded by the sum of the elements in the set. Note that the complete search algorithm given earlier also runs in $O(n2^n)$. Although the time complexities of both algorithms are identical, the dynamic programming one is generally faster due to its use of optimal substructure and overlapping subproblems. In fact, this is the fastest known runtime of any classical algorithm for the Subset Sum Problem.

C. OVERVIEW OF QUANTUM COMPUTATION

A quantum computer is one that manipulates the components of its internal state by exploiting the principles of quantum mechanics (Mermin). Though both classical and quantum computers share the elementary functionality of Turing machines, quantum computers utilize special properties of physical systems that are precisely why quantum algorithms are so efficient (Rosen). Thus, a basic understanding of quantum computational theory is necessary prior to implementing a quantum search algorithm.

http://scholarcommons.usf.edu/ujmm/vol4/iss2/2
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.2

In classical computers, all operations and data storage are dependent upon the bit, the fundamental computational unit. It can assume only one of two discrete states, 1 or 0, which can be interpreted by the computer as "on" or "off," Boolean values of true or false, or as binary numbers when multiple bits are strung together. Analogously, quantum computers rely on quantum bits, or qubits, for data manipulation. Qubits can also be associated with the aforementioned states, conventionally written in Dirac notation as $|1\rangle$ and $|0\rangle$ (Viamontes, Markov and Hayes).

Whereas classical bits can be stored using high and low voltages in computer chips, qubits must have a representation grounded in the physical world in order to be properly manipulated. For example, using a hydrogen atom, the states $|1\rangle$ and $|0\rangle$ corresponds to an electron in the excited state and in the ground state respectively (Dasgupta, Papadimitriou and Vazirani). Abstractly, however, the linearity of quantum mechanics allows linear algebra to aptly model qubits as vectors and bit manipulations as unitary transformations.

One of the major differences between qubits and classical bits is that the former are subject to the superposition principle, which states that a quantum system can exist in a linear superposition of all possible states (Dasgupta, Papadimitriou and Vazirani). As such, a quantum state $|\psi\rangle$ of a single qubit can be represented as a unit vector given by a linear combination of the set of all states, known as the computational basis (Mermin):

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$$

where α_i, the amplitude of a state's mechanical wave function, is an arbitrary complex number such that $|\psi\rangle$ is normalized, i.e. (Mermin)

$$|\alpha_0|^2 + |\alpha_1|^2 = 1.$$

The quantum state of a multiple qubit system is the tensor product of the individual qubit states; for instance, given two qubits, the quantum system is characterized by a superposition of four computational basis states as (Mermin)

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

where the first number in each bit string corresponds to the first qubit and the second number corresponds to the second qubit (this bit string is commonly replaced with its decimal representation). The state vector is also normalized, following the condition (Mermin)

$$|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1.$$

In general, an n-qubit system yields a computational basis of 2^n states, where the sum of the squared magnitudes of the amplitudes equals one (Mermin):

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x|x\rangle,$$

$$\sum_{x\in\{0,1\}^n} |\alpha_x|^2 = 1$$

The superposition principle is one of the factors contributing to the immense computational power of a quantum computer. First of all, a quantum system is uniquely defined by the 2^n amplitudes of n qubits; for moderately large values of n, a classical computer would require an exponentially large amount of memory, but a quantum computer needs only n qubits to store the amplitudes of the computational basis states (Dasgupta, Papadimitriou and Vazirani). Secondly, since quantum mechanics conforms to linear algebra quite well, one operation on an n-qubit quantum state is equivalent to 2^n simultaneous individual operations on the computational basis. This phenomenon, known as quantum parallelism, is carried out by the properties of linear transformations and allows for exponentially large number of operations to be executed instantaneously, another reason why quantum algorithms outperform their classical

http://scholarcommons.usf.edu/ujmm/vol4/iss2/2
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.2

counterparts (Viamontes, Markov and Hayes). Entanglement is another phenomenon that is vital to many applications of quantum computing, but it will not be discussed here since it holds no relevance to Grover's algorithm. As such, it will be assumed that the individual states of qubits for the Subset Sum Problem are not entangled in a quantum system.

Such an improvement in memory and speed comes with a drawback, however. Although a qubit is in a linear combination of multiple states, the only way to extract information about any of them is to make a measurement, the act of observing the state of the qubit (Dasgupta, Papadimitriou and Vazirani). Unfortunately, merely observing one qubit collapses its mechanical wave function, forcing its state from the superposition $\alpha_0|0\rangle + \alpha_1|1\rangle$ to a single computational basis state, either $|0\rangle$ or $|1\rangle$. Similarly, measuring an n-qubit system requires the measurement of each qubit, producing only one state out of a computational basis of 2^n states. By Born's Rule, the probability of a particular state arising from a measurement of a system is (Mermin)

$$p(x) = |\alpha_x|^2.$$

Note that the normalization condition for the state vector $|\psi\rangle$ is consistent with Born's Rule, as the magnitude of the vector and the sum of all probabilities is equal to exactly one.

For a given set of qubits, the amplitudes cannot be known, for they are arbitrary complex numbers; in fact, the system reduces into a single state with an effectively unknown probability, all amplitudes erased due to measurement. Quantum computers are not deterred, however, as there remains a way to gather information from qubits.

Since the outcome of a specific state is probabilistic, it is inherent that quantum algorithms are not deterministic. Classical computers pass bits through electrical circuits and logic gates in order to manipulate bits, providing the basic functionality algorithms need in order to operate. Likewise, the general procedure for quantum algorithms is to obtain qubits in a

known state by measurement and pass them through elementary quantum gates, manipulating the qubits by enhancing, damping, or inverting amplitudes with the goal of maximizing the amplitudes—and hence the probability—of desired states while minimizing those of undesirable ones (Dasgupta, Papadimitriou and Vazirani). Thus, the goal of the quantum algorithm is to generate the correct output upon measurement with high probability.

D. QUANTUM GATES

Bit operations in classical computers occur by passing bits through logic gates to be transformed according to their current states. These gates are physical implementations of the logical operators AND, OR, XOR, and NOT, and by linking together a series of logic gates, any computation can theoretically be achieved (Berman, Doolen and Mainieri). Quantum computers also rely on similar constructions known as quantum gates for bit manipulation. The linearity of quantum mechanics allows for these quantum gates to be modeled as a linear transformation on the state vector of the system. The most significant difference between classical logic gates and quantum ones is the requirement for the matrix representation of any transformation of a quantum gate to be unitary, that is, the Hermitian conjugate of the matrix is also the inverse (Nielson and Chuang):

$$AA^{\dagger} = A^{\dagger}A = I$$

In order for quantum computation to be physically realized, all transformations must obey the conditions stipulated by the Schrödinger equation (Berman, Doolen and Mainieri). First, a solution to the Schrödinger equation for a quantum system at an arbitrary moment in time can be derived from any other known solution (Berman, Doolen and Mainieri). Consequently, quantum gates must be reversible, as it allows for the qubits to reflect the state of the quantum computer at any point between computations as long as measurements are not made. Secondly,

http://scholarcommons.usf.edu/ujmm/vol4/iss2/2
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.2

for efficient computation, the Hamiltonian of the system should be time-independent (Berman, Doolen and Mainieri). According to Landauer's Principle, a computer releases energy into the environment upon the erasure of a single bit (Nielson and Chuang). Reversible computation prevents this flow of energy, preserving the Hamiltonian in turn, because erasing bits is unnecessary when past states can be reconstructed by undoing operations. Thus, all unitary matrices satisfy these conditions since they are invertible and norm-preserving. Classical logic gates fail to meet the criteria, mainly because they are not invertible. For example, the XOR operation takes as input two bits but outputs only one bit; with a loss of information, it is impossible to derive the state of each input bit given the state of the output bit.

Reversible counterparts of the logic gates exist for quantum computation, so it is possible to simulate any classical operation with a quantum computer (Berman, Doolen and Mainieri). Furthermore, it has been shown that any 2×2 unitary matrix can be decomposed as (Nielson and Chuang)

$$U = e^{i\alpha}\begin{bmatrix} e^{-\frac{i\beta}{2}} & 0 \\ 0 & e^{\frac{i\beta}{2}} \end{bmatrix}\begin{bmatrix} \cos\frac{\gamma}{2} & \sin\frac{\gamma}{2} \\ \sin\frac{\gamma}{2} & \cos\frac{\gamma}{2} \end{bmatrix}\begin{bmatrix} e^{-\frac{i\delta}{2}} & 0 \\ 0 & e^{\frac{i\delta}{2}} \end{bmatrix}$$

where $\alpha, \beta, \gamma,$ and δ are real numbers, allowing for an arbitrary construction for a single qubit quantum gate provided with the correct parameters. This result applies not only for single qubit gates, however, as it has been shown that any multiple qubit gate can be constructed using one- and two-qubit gates (Nielson and Chuang). As a result, the discussion of Grover's algorithm will not be overly concerned with the physical possibility of quantum gates as long as it is known that any gate can be created. Although arbitrary gates may be constructed, there are two specific quantum gates used extensively in computation, especially in Grover's algorithm: the standard unitary transform and the Hadamard transform.

1) The Unitary Transform: The standard unitary transform used by most quantum algorithms is given by (Mermin)

$$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle,$$

where $\oplus$ represents addition modulo 2 and $f(x)$ is some auxiliary function, usually the oracle.

U_f accepts a single vector in the form of a tensor product between $|x\rangle$ and $|y\rangle$, which are known as the input and output registers respectively (Mermin). The input register is typically the state of the quantum system, and the output register is some other *n*-qubit state that varies with respect to the specific application of U_f. Furthermore, it is clear that the standard unitary transform is its own inverse. The most significant feature of U_f is that it allows the output register to record information about $|x\rangle$ via $f(x)$. In this way, it is possible to gain some information about $|x\rangle$ without collapsing it by measuring $|y \oplus f(x)\rangle$ instead, leaving $|x\rangle$ available for further processing (Mermin).

2) The Hadamard Transform: One of the basic quantum gates, the Hadamard transform acting on a single qubit yields the following (Dasgupta, Papadimitriou and Vazirani):

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Like U_f, H is its own inverse. More importantly, the properties of tensor products allows for multiple Hadamard gates to act on multiple qubits. It can be easily verified that (Mermin)

$$H^{\otimes n}|0\rangle_n = \frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle,$$

http://scholarcommons.usf.edu/ujmm/vol4/iss2/2
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.2

otherwise known as the equal superposition of the computational basis. Qubits in the $|0\rangle$ or $|1\rangle$ state are easily obtainable by measurement, so $H^{\otimes n}$ is extremely useful in generating a uniform qubit state when n qubits only in the $|0\rangle$ state are available.

E. GROVER'S ALGORITHM

Grover's algorithm is a general quantum search technique that can find an element within an unsorted search space with extremely high probability. Since NP-complete problems are defined to be search problems, Grover's algorithm can be applied to the Subset Sum Problem with several adjustments and a slight computational overhead. For ease of explanation, it will be assumed for now that there is only one solution to the Subset Sum Problem, as the presence of additional solutions affects the algorithm only by enhancing the probability of success.

1) The Oracle: The oracle, also known as the black-box function, is a special function whose implementation is unknown but can be assumed to execute a specific duty reliably in polynomial time (Viamontes, Markov and Hayes). Oracles are common in quantum algorithms, as they help to simplify the problem abstraction and divert attention away from the uninteresting technicalities of constructing such a basic function. For completeness, however, the structure of the oracle will be discussed. In Grover's algorithm, the oracle's purpose is to recognize if a given state is the one sought after. It is important to clarify that this does not imply that the oracle can find solutions by itself; rather, the oracle can only test possible solutions in polynomial time. Furthermore, solution states are trapped within the superposition, so the other operations in Grover's algorithm are required to retrieve the state identified by the oracle.

For the context of the Subset Sum Problem, define a bijection between subsets of elements and the states of qubits such that the index of a qubit corresponds with the index of an

element in the set: if the *i*th qubit of $|x\rangle$ is $|1\rangle$, include x_i in the subset, but do not use x_i if the qubit is $|0\rangle$. Thus, a one-to-one and onto relation is created, so the oracle can be implemented through this meaningful bijection.

It is necessary to define another reference for the input size as $N = 2^{\lceil \log n \rceil}$ (by convention, the logarithm is in base 2). That is, N represents the smallest power of 2 greater than n. The number of computational basis states must be a power of 2, so this refinement of N is required in order to record all the correspondences whenever n is not already a power of 2. Of course, this means that there may be more computational basis states than the bijection should allow, but this need not be a hindrance because the oracle can identify and skip any invalid state.

The oracle $f(x)$ returns 1 if *x* matches the desired state a and returns 0 otherwise:

$$f(x) = \begin{cases} 1 & if\ x = a \\ 0 & if\ x \neq a \end{cases}$$

where a is the qubit state whose bit string representation corresponds to a solution subset according to the bijection. In order to determine whether a given state is indeed the correct bit string representation, the oracle will follow the bijection, calculate the corresponding subset sum, and then compare it to the target integer. Since this procedure is essentially the addition of multiple elements, the time complexity of the oracle is $O(n)$.

By using the standard unitary transformation $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$,

the oracle flips the output register if x represents the subset of elements whose sum is T. By quantum parallelism, applying one unitary transformation on a state vector is equivalent to transforming all 2^n computational basis states, evidence already that Grover's algorithm is much more efficient than the classical algorithm. However, this operation is still insufficient, as U_f must somehow distinguish the correct state from the incorrect ones, but this unitary

http://scholarcommons.usf.edu/ujmm/vol4/iss2/2
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.2

transformation leaves the input register invariant. One way to mark the desired element is to apply a phase inversion to the computational basis state by multiplying its amplitude by -1 while leaving all other states untouched (Morsch). In order to accomplish this, consider the Hadamard transform on the state $|1\rangle$,

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

When a bit flip is applied,

$$\widetilde{H|1\rangle} = \frac{1}{\sqrt{2}}(|1\rangle - |0\rangle) = -\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = -H|1\rangle,$$

which is equivalent to multiplying the qubit by -1. Upon further inspection of the output register $y = H|1\rangle$, if the oracle returns a 1, then the tensor product of the input and output registers becomes negative, and it is left alone when the oracle returns 0. Thus, the unitary transform can be rewritten as (Mermin)

$$U_f|x\rangle \otimes H|1\rangle = (-1)^{f(x)}|x\rangle \otimes H|1\rangle.$$

A conceptual implementation of the oracle is now complete. As for its physical implementation, it has been discussed previously that any quantum gate can be physically realized, so it will not be of concern here.

2) The Grover Iteration: Grover's algorithm begins by setting $|\psi\rangle$ to be the equal superposition state obtained by passing *n* qubits in the $|0\rangle$ state through the Hadamard gate (Mermin):

$$|\psi\rangle = H^{\otimes n}|0\rangle_n = \frac{1}{N^{1/2}}\sum_{x=0}^{N-1}|x\rangle.$$

Next, the state vector is passed through the standard unitary transformation with $|\psi\rangle$ in the input register and $H|1\rangle$ in the output register, applying a phase inversion to the amplitudes of the desired computational basis states:

$$U_f|\psi\rangle \otimes H|1\rangle = (-1)^{f(x)}|\psi\rangle \otimes H|1\rangle.$$

As discussed earlier, the ancillary qubit $H|1\rangle$ is tensored with the input register in order for the phase inversion to work properly; however, since this particular qubit is not required for any other operation within Grover's algorithm, it will be omitted from further description. Note that since the unitary transform indicates a proper tensor product between the input and output registers, the state vector $|\psi\rangle$ does not suffer from entanglement, so a measurement can be taken regardless of the presence of $H|1\rangle$.

Afterwards, a quantum gate $D = H^{\otimes n}CH^{\otimes n}$ is applied, where C is the controlled phase gate (Nielson and Chuang). The action of C is to invert the phase of every computational basis state save for $|0\rangle$, which can be described by $C = 2|0\rangle_n\langle 0|_n - I$ (Nielson and Chuang). The net effect of applying Hadamard transforms before and after the controlled phase gate allows for D to be rewritten as the diffusion matrix (Nielson and Chuang)

$$D = H^{\otimes n}CH^{\otimes n} = 2|\phi\rangle\langle\phi| - I,$$

where $|\phi\rangle$ is the equal superposition state that remains invariant within the gate. Essentially, the diffusion matrix represents the inversion about the mean operation (Grover). For any given amplitude α of a computational basis state and the mean of all amplitudes μ, the amplitude is below the mean by exactly $\mu - \alpha$. The diffusion matrix replaces the amplitude α with a new quantity α' that is above the mean by the same amount that α was below the mean, or

$$\alpha_i' = \mu + (\mu - \alpha_i) = 2\mu - \alpha_i,$$

thereby inverting the amplitude of the state $|i\rangle$ about the mean (Morsch).

http://scholarcommons.usf.edu/ujmm/vol4/iss2/2
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.2

The inversion about the mean operation is essential for the magnification of the desired amplitude and hence the probability of correct output. Since the oracle function inverted the phase of the desired state, it is farther below the mean amplitude compared to the other amplitudes. After applying the diffusion matrix, the desired amplitude will be farther above the mean than the other amplitudes, which will have decreased. For example, in Figure 1, the amplitudes of four different states are represented by vertical bars, and C is the farthest away from the average line due to its inverted phase. However, after the inversion about the mean operation, the amplitudes are above the mean by exactly the same amount they were once below it; since C was the farthest below, it now becomes the farthest above.

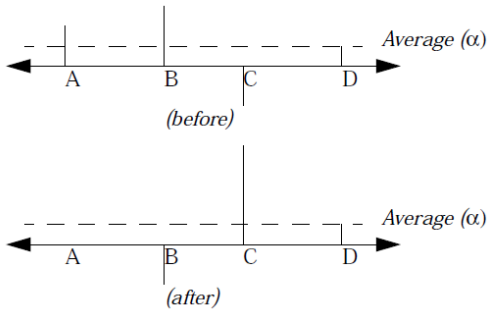

Figure 1: The inversion about the mean operation on the amplitudes of four states (Grover).

The oracle function and the inversion about the mean operation together constitute the Grover iteration, the part of the algorithm that must be repeated several times in order to improve probability of success. As such, the Grover iteration $G = DU_f$ is applied repeatedly on the initial input $|\psi\rangle = H^{\otimes n}|0\rangle_n$ (Mermin). Figure 2 demonstrates the results of the first Grover iteration on the amplitudes of the computational basis states. For up to a certain number of iterations, the amplitude of the desired state will continuously increase while the others approach 0. At this point, a measurement is carried out to obtain one possible solution state $|\psi'\rangle$, and due to repeated use of *G*, the probability of observing a desired computational basis state is close to 1.

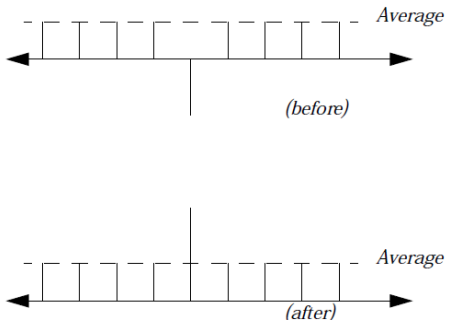

Figure 2: Single call of the Grover iteration on the equal superposition state (Grover).

Since the Subset Sum Problem is not concerned with what particular subset is the solution, for there could be multiple subsets, the final unitary transform is applied:

$$U_f|\psi'\rangle|0\rangle = |\psi'\rangle|f(\psi')\rangle.$$

Simply measuring the output register will reveal whether $|\psi'\rangle$ is the correct solution: if the measurement produces $|1\rangle$, then the state is correct, or else $|0\rangle$ indicates a faulty solution.

3) Geometric Interpretation: Grover's algorithm can be visualized geometrically as linear transformations acting upon vectors in space, providing a medium for deriving the time complexity of this quantum algorithm. First of all, one must realize that the computational basis is in fact an orthonormal basis of a Hilbert space, as any arbitrary state vector within the vector space may be created with complex linear combinations of the computational basis states. Given a computational basis of N states, let there be M possible solutions to the Subset Sum Problem. Furthermore, let A be the set of all computational basis states and B be the subset representing all desirable states. Two new vectors can be created with linear combinations of the vectors in B and its complement (Nielson and Chuang):

$$|\alpha\rangle = \frac{1}{\sqrt{N-M}} \sum_{a \in B^c} |a\rangle,$$

http://scholarcommons.usf.edu/ujmm/vol4/iss2/2
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.2

$$|\beta\rangle = \frac{1}{\sqrt{M}} \sum_{b \in B} |b\rangle.$$

In other words, $|\alpha\rangle$ is the linear combination of all undesirable states and $|\beta\rangle$ is the linear combination of all desirable ones, both of which have been normalized. Therefore, the initial input state of equal superposition

$$|\psi_0\rangle = H^{\otimes n}|0\rangle_n = \frac{1}{N^{1/2}} \sum_{x=0}^{N-1} |x\rangle$$

can be rewritten as a linear combination of $|\alpha\rangle$ and $|\beta\rangle$, normalizing once more (Nielson and Chuang):

$$|\psi_0\rangle = \sqrt{\frac{N-M}{N}}\,|\alpha\rangle + \sqrt{\frac{M}{N}}\,|\beta\rangle.$$

Note that $|\alpha\rangle$, $|\beta\rangle$, and $|\psi_0\rangle$ reside within the same vector space, as they are normalized and are linear combinations of the basis vectors.

The computational basis states are orthonormal, as each is normalized and any pair of vectors is orthogonal. Since $|\alpha\rangle$ and $|\beta\rangle$ are linear combinations from disjoint sets of an orthonormal basis, they are orthonormal as well. Consequently, $|\alpha\rangle$ and $|\beta\rangle$ together span a plane, forming the basis of a two dimensional Hilbert space. The most important corollary to this realization is that state vectors may be rewritten as linear combinations of arbitrarily defined computational basis states; that is, through a change of basis, qubits may be subjected to measurement with respect to an entirely new yet equivalent basis (Nielson and Chuang). As a result, it is fitting to describe the vectors $|\alpha\rangle$, $|\beta\rangle$, and $|\psi_0\rangle$ as equivalent to

$$|\alpha\rangle \equiv |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

$$|\beta\rangle \equiv |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

$$|\psi_0\rangle \equiv \sqrt{\frac{N-M}{N}}|0\rangle + \sqrt{\frac{M}{N}}|1\rangle = \begin{pmatrix} \sqrt{(N-M)/N} \\ \sqrt{M/N} \end{pmatrix},$$

where the column vectors for $|0\rangle$ and $|1\rangle$ are defined by Dirac notation. Therefore, the entire quantum system can be considered to take place within a plane (Nielson and Chuang).

$|\beta\rangle$ and $|\psi_0\rangle$ are very nearly orthogonal, but $|\beta\rangle$ and $|\alpha\rangle$ are entirely orthogonal, so $|\psi_0\rangle$ and $|\alpha\rangle$ are separated by a very small angle θ, which can be found by taking the inner product (Mermin)

$$\langle\beta|\psi_0\rangle = \cos\left(\frac{\pi}{2} - \theta\right) = \sin\theta = \sqrt{\frac{M}{N}} \approx \theta,$$

where the last approximation is highly accurate for large *N* by the Small Angle Approximation.

As shown in Figure 3, Grover's algorithm applies two reflections in the plane, the net result of which is a rotation. First, the unitary transformation with the oracle function reflects the initial vector across $|\alpha\rangle$ because the phase inversion on $|\beta\rangle$ reverses the direction of the corresponding component in $|\psi_0\rangle$:

$$U_f\left(\sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}|\beta\rangle\right) = \sqrt{\frac{N-M}{N}}|\alpha\rangle - \sqrt{\frac{M}{N}}|\beta\rangle.$$

After U_f, $|\psi_0\rangle$ will be θ below $|\alpha\rangle$, displaced a total of 2θ. Similarly, the inversion about the mean operation $D = 2|\phi\rangle\langle\phi| - I$ reflects $U_f|\psi_0\rangle$ across $|\phi\rangle$, which happens to be equivalent to $|\psi_0\rangle$ in the first Grover iteration (Mermin). D brings $U_f|\psi_0\rangle$ towards $|\beta\rangle$ by 4θ, creating a net rotation of 2θ radians for a new state $DU_f|\psi_0\rangle = |\psi_1\rangle$. Therefore, each application of the Grover iteration rotates the state vector towards $|\beta\rangle$, the superposition of all desirable states, by 2θ (Nielson and Chuang). After enough iterations, the final state vector is extremely close to $|\beta\rangle$, so the amplitude of $|\alpha\rangle$ is drastically decreased while the amplitude of $|\beta\rangle$ experiences a

http://scholarcommons.usf.edu/ujmm/vol4/iss2/2
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.2

corresponding increase. By Born's Rule, the probability of observing $|\beta\rangle$ is close to one, and by the original definition of $|\beta\rangle$ being the linear combination of all desirable states, any one of those is equally likely to appear.

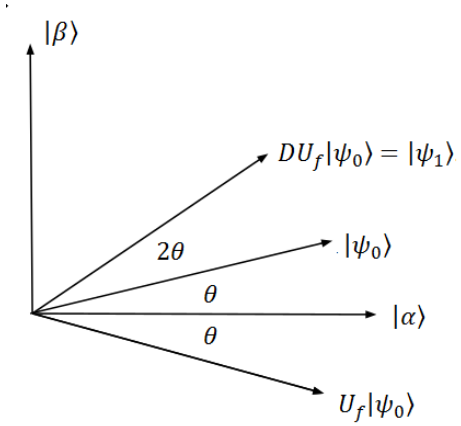

Figure 3: Rotation of state vector in a two dimensional Hilbert space (not drawn to scale) (Mermin).

4) Time Complexity of the Quantum Algorithm: Ideally, Grover's algorithm will rotate $|\psi_0\rangle$ until it is almost nearly $|\beta\rangle$ to maximize the probability that one of the desirable states will be selected, so an optimal number of iterations must be calculated. As mentioned before, $|\psi_0\rangle$ and $|\beta\rangle$ are almost orthogonal, so the state vector must be rotated by approximately $\frac{\pi}{2}$ radians. Since the Grover iteration rotates by 2θ radians each time, the quantum computer must apply G about $\frac{\frac{\pi}{2}}{2\theta} = \frac{\pi}{4\theta}$ times (Vazirani). In order to attain proper time complexity by order of growth analysis, this quantity must involve the input size. From earlier,

$$\langle\beta|\psi_0\rangle = \sqrt{M/N} \approx \theta,$$

so it follows that

$$\frac{\pi}{4\theta} = \frac{\pi}{4}\sqrt{N/M} = O\left(\sqrt{N/M}\right),$$

or for a worst-bound case when $M = 1$,

$$O\left(\sqrt{N/M}\right) = O(2^{n/2})$$

is the optimal number of Grover iterations, and hence oracle calls, of Grover's algorithm (Grover).

DISCUSSION

Since classical and quantum computation differ fundamentally by the type of operations carried out, the time complexity of quantum algorithms is conventionally written in terms of the number of oracle calls rather than operations. The time complexity of the dynamic programming algorithm given previously will also be converted in order to give an accurate comparison of efficiency.

The dynamic programming algorithm does not use oracle calls, but it is still possible to rewrite its order of growth with respect to the time complexity of the oracle. As discussed, the oracle essentially sums elements in the list according to the bit string passed to it, so its performance is $O(n)$. The number of linear time operations used by the dynamic programming algorithm is $\frac{n2^n}{n} = 2^n$, so $O(2^n)$ oracle calls are required. However, Grover's algorithm uses $O(2^{n/2})$ oracle calls, so it significantly outperforms the classical approach by a factor of $2^{n/2}$, a quadratic reduction in time. This is generally true for any specific application of Grover's algorithm: the runtime of a quantum search is the square root of the runtime of the analogous classical search (Nielson and Chuang).

One of the most promising applications of quantum search is to resolve the exponential time complexity that is characteristic of NP-complete problems. Since Grover's algorithm provides a quadratic improvement in speed, one may conjecture that an even faster quantum algorithm, perhaps a polynomial one, may exist. Unfortunately, it has been proven that Grover's

http://scholarcommons.usf.edu/ujmm/vol4/iss2/2
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.2

algorithm is optimal, so the fastest time complexity of any search algorithm, classical or quantum, is $O(2^{n/2})$ (Zalka). Moreover, the NP-complete class is defined to be a set of search problems that reduce to one another, yet the fastest possible search algorithm is still exponential, which severely diminishes the hope that NP-complete problems do have a polynomial time solution. However, there exist several alternatives, though not promising, to consider.

It is widely believed that NP-complete problems are intractable because there is no exploitable organization in the underlying structure of the search space (Nielson and Chuang). In fact, it is precisely this property that makes certain classical deterministic algorithms so efficient; for instance, Prim's algorithm for finding the minimum spanning tree in a graph operates in polynomial time since the solution satisfies the greedy property of optimizing locally to optimize globally. As such, it may be possible to expedite solutions to the NP-complete class by reexamining the search space in novel ways. For example, the dynamic programming algorithm for the Subset Sum Problem presented earlier runs in pseudopolynomial time by utilizing optimal substructure and overlapping subproblems. Though this approach was not sufficiently fast, it demonstrates that there could be some property of the Subset Sum Problem, and possibly the entire NP-complete class, not yet observed that can be used as the basis of an even faster algorithm.

Another possibility to consider is the application of reductions. Finding the prime factorization of an integer is a famously hard problem (though not NP-complete) that has no known polynomial time solution in the classical realm, but quantum computation offers one. Shor's algorithm, a quantum algorithm, reduces the problem of finding prime factors into the problem of identifying periodicity, which can be solved with the Fourier transform and some number theoretic insights (Morsch). Similarly, reductions may help bring known polynomial

time solutions to the NP-complete problems. Of course, the largest and most overwhelming obstacle to this consideration by far is that for an NP-complete problem to reduce to P, it would require the proof of $P = NP$, a puzzle that has eluded generations of computer scientists. Still, much of computational complexity theory analyzes problems in respect to classical algorithms, so perhaps reductions of the NP-complete class may be possible with advances in quantum computation.

Since the search for a polynomial time algorithm for NP-complete problems is bleak, other approaches must be used for the time being. Despite its exponential growth, Grover's algorithm is optimal and in fact has just been shown to solve the Subset Sum Problem, and since all NP-complete problems reduce to one another, quantum search certainly works on the entire complexity class. Therefore, it is not only feasible but also more efficient to apply quantum search techniques on NP-complete problems, especially when exact solutions are required, rather than relying on heuristics and approximation algorithms.

CONCLUSION AND RECOMMENDATIONS

The Subset Sum Problem remains part of the NP-complete class, but Grover's algorithm achieves a notable improvement in time complexity over any known classical algorithm. Furthermore, by reductions and slight computational overhead, quantum search can be applied to any instance of NP-completeness, evincing the possibility of diverse application across fields as well, such as in primality testing or cryptography. Although the physical aspects of quantum computers were not discussed in depth, they do present significant but definitely tractable challenges for engineers. Nevertheless, it is exciting to consider that quantum computational theory may yield in the near future new algorithms inconceivable in a classical domain yet innovative and possibly revolutionary for all of computer science.

http://scholarcommons.usf.edu/ujmm/vol4/iss2/2
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.2

NOMENCLATURE

Symbol	Description
$O(f(x))$	Big-O notation. For an algorithm with input data size x and processing time $g(x)$, $O(f(x))$ indicates that $g(x)$ is bounded above by $f(x)$.
$\|a\rangle$	Ket; in Dirac notation, a column vector. If a is a nonnegative integer, its binary form represents a tensor product of a sequence of $\|0\rangle$ and $\|1\rangle$ kets.
$\|\psi\rangle$	Vector representing the state of a quantum system; i.e. a complex linear combination of possible discrete states the unmeasured quantum system can assume.
$\|0\rangle, \|1\rangle$	Defined in Dirac notation as the column vectors $\binom{1}{0}$ and $\binom{0}{1}$ respectively.
$\langle a\|$	Bra; or a row vector.
$\langle a\|b\rangle$	Inner product between $\|a\rangle$ and $\|b\rangle$.
$\|a\rangle\langle b\|$	Outer product between $\|a\rangle$ and $\|b\rangle$.
$\|a\rangle \otimes \|b\rangle$, $\|a\rangle\|b\rangle, \|ab\rangle$	Tensor product between $\|a\rangle$ and $\|b\rangle$.
$A^{\otimes n}$	n-fold tensor product of A with itself.
$\oplus$	XOR operator; i.e. bitwise addition modulo 2.

REFERENCES

Berman, Gennady P., et al. Introduction to Quantum Computers. Singapore: World Scientific, 1999. 38, 85-86.

Dasgupta, Sanjoy, Christos H. Papadimitriou and Umesh Vazirani. Algorithms. Boston: McGraw-Hill Higher Education, 2008. 243, 297-298, 301, 307.

Grover, Lov K. "Quantum Mechanics Helps in Searching for a Needle in a Haystack." Physical Review Letters 79.2 (1997): 325-328.

Larson, Ron, Robert Hostetler and Bruce Edwards. Calculus. 8th Edition. Boston, MA: Houghton Mifflin Company, 2005.

Mermin, David N. Quantum Computer Science: An Introduction. Cambridge: Cambridge University Press, 2007. 1, 17, 24, 37-38, 89-94.

Morsch, Oliver. Quantum Bits and Quantum Secrets: How Quantum Physics is Revolutionizing Codes and Computers. Weinheim: Wiley-VCH, 2008. 96, 109.

Neapolitan, Richard E. and Kumarss Naimipour. Foundations of Algorithms: Using C++ Pseudocode. Sudbury: Jones and Bartlett Publ., 1998. 194.

Nielson, Michael A. and Isaac L. Chuang. Quantum Computation and Quantum Information. Cambridge: Cambridge University Press, 2000. 29, 62, 153, 249-253.

Rosen, Kenneth H. Discrete Mathematics and Its Applications. Boston: McGraw-Hill, 2003. 781.

Vazirani, Umesh. Chem/CS/Phys191: Qubits, Quantum Mechanics, and Computers. n.d. 24 June 2012 <http://www-inst.eecs.berkeley.edu/~cs191/sp12/>.

Viamontes, George F., Igor L. Markov and John P. Hayes. "Is Quantum Search Practical?" Computing in Science and Engineering 7.3 (2005): 62-70.

Zalka, Christof. "Grover's quantum searching algorithm is optimal." Physical Review A 60.4 (1999): 2746-2751.

http://scholarcommons.usf.edu/ujmm/vol4/iss2/2
DOI: http://dx.doi.org/10.5038/2326-3652.4.2.2