We present a quantum algorithm to compute the number of solutions of the (constrained) number partitioning problem. We show that on a quantum computer, this algorithm solves the number partitioning problem in polynomial time and space.

PACS numbers: 03.67.Lx, 89.70.+c

The discovery of quantum algorithms that, when executed on a quantum computer (QC), give significant speedup over their classical counterparts [1,2] has given strong impetus to recent developments in the field of quantum computation. In this contribution, we present a new quantum algorithm that fully exploits the potential power of a QC. It solves a basic problem of combinatorial optimization: The number partitioning problem. We show that on a quantum computer, this algorithm solves the number partitioning problem in polynomial time if the sum of all \(a\) is odd (even). In this paper, we will use the latter formulation.

Number partitioning is one of Garey and Johnson’s six basic NP-complete problems [4]. It is a key problem in the theory of computational complexity and has a number of important practical applications such as job scheduling, task distribution on multiprocessor machines, VLSI circuit design to name a few.

The NPP can be solved by dynamic programming, in a time bounded by a low order polynomial in \(nB\) [4]. For a given instance of \(A = \{a_1, \ldots, a_n\}\), we may encode the whole problem using only \(n \log_2 B\) bits. As \(nB\) is not bounded by any polynomial of the input size \(n \log_2 B\), the dynamic programming algorithm does not solve the NPP in polynomial time [4].

In practice, the computation time to solve a NPP depends on the number of bits \(b = \log_2 B\) needed to represent the integers \(a_j\) and \(B\). Numerical simulations using random instances of \(A\) show that the solution time grows exponentially with \(n\) for \(n \ll b\) and polynomially for \(n \gg b\) [6–9]. For random instances \(A\), the NPP can be mapped onto a hard problem of statistical mechanics, namely, that of finding the ground state of an infinite-range Ising spin glass [10–12]. The transition from the computationally “hard” (exponential) to “easy” (polynomial) has been related to the phase transition in the statistical mechanical system [10,12].

For certain applications, there may be additional constraints on the partitioning of the set \(A\). A common one is to fix the difference \(C\) between the number of elements in \(A_1\) and \(A_2\): \(C = \sum_{a_j \in A_1} 1 - \sum_{a_j \in A_2} 1\). For instance, if \(C = 0\) we ask if there is a partitioning such that the number of elements in \(A_1\) and \(A_2\) is the same.

The potential power of a QC stems from the fact that a QC operates on superpositions of states [13–19]. The interference of these states allows exponentially many computations to be done in parallel [13–19]. A quantum algorithm consists of a sequence of unitary transformations that change the state of the QC [13–19]. Therefore to solve a NPP on a QC, we first have to develop an algorithm that can be expressed entirely in terms of unitary operations.

A generic \(n\)-qubit QC can be modeled by a collection of \(n\) two-state systems, represented by \(n\) Pauli spin matrices \(\{\vec{\sigma}_1, \ldots, \vec{\sigma}_n\}\) [13–19]. The two eigenstates of \(\sigma_j^z\) will be denoted by \(|+\rangle_j\) and \(|-\rangle_j\), corresponding to the states \(|0\rangle_j\) and \(|1\rangle_j\) of the \(j\)-th qubit respectively. The eigenvalues corresponding to \(|+\rangle_j\) and \(|-\rangle_j\) are \(S_j = +1\) and \(S_j = -1\). They can be used to represent a partitioning of \(A\): We assign \(a_j\) to \(A_1\) (\(A_2\)) if \(S_j = 1\) (\(S_j = -1\)). If we can find a set \(\{S_1, \ldots, S_n\}\) such that \(E = \Delta - \sum_{j=1}^n a_j S_j = 0\), we have found one solution of the NPP. The numbers \(E\) are the eigenvalues of the Hamiltonian \(H = \Delta - \sum_{j=1}^n a_j \sigma_j^z\). Thus a solution of the NPP corresponds to an eigenstate of \(H\) with energy zero. This is one key to the construction of a polynomial-time quantum algorithm to solve NPP’s on a QC.

It is known that the most simple class of spin system, i.e., those involving interactions of the Ising type only, can be used to build universal QC’s [14,18,20]. The Hamiltonian \(H = \Delta - \sum_{j=1}^n a_j \sigma_j^z\) describes \(n\) non-interacting spins in external fields represented by the \(a_j\)’s and is of...
the Ising type. We will use this Hamiltonian to define the time evolution of the QC, i.e. the quantum algorithm that solves NPP’s.

The second key to the construction of the quantum algorithm is the observation that the number of solutions \( n_s \) of a NPP is given by

\[
\frac{1}{M} \sum_{m=0}^{M-1} \text{Tr} \, e^{-2\pi i mH/M},
\]

where \( M \equiv B + \Delta + 1 \) and \( \text{Tr} \, U \) denotes the trace of the matrix \( U \) [21]. Indeed, using the representation that diagonalizes the spin operators \( \sigma_j^\pm \), we find

\[
n_s = \sum_{\{S_j=\pm 1\}} \left( \frac{1}{M} \sum_{m=0}^{M-1} \exp \left[ \frac{2\pi i m}{M} \left( \sum_{j=1}^n a_j S_j - \Delta \right) \right] \right)
= \sum_{\{S_j=\pm 1\}} \left( 1 - \exp \left[ \frac{2\pi i \sum_{j=1}^n a_j S_j - \Delta}{M} \right] \right),
\]

(2)

where \( |\Delta - \sum_{j=1}^n a_j S_j| < M \) for any choice of \( \{S_j\} \), the sum over \( m \) in (2) will be zero unless \( \Delta - \sum_{j=1}^n a_j S_j = 0 \), in which case the configuration \( \{S_1, \ldots, S_n\} \) is a solution of the number partitioning problem (note that there can be exponentially many solutions, for instance if all the \( a_j \)'s are the same). Performing the sum over all spin configurations as indicated in (2), it follows immediately that \( n_s \) is the number of solutions of the NPP. Note that (2) gives the number of solutions of a NPP, which is more than just a yes or no answer to the question if a partition of \( A \) exists [4].

Formally (1) is the density of states at zero energy of the physical system described by Hamiltonian \( H \). Elsewhere we have shown that for a large class of models \( H \), the density of states can be calculated efficiently on a QC [22]. The algorithm presented below, although related to the one described in [22], is specifically tuned to solve NPP’s.

The equivalence of (2) and the solution of the NPP can also be shown by explicit calculation of the trace over all spin configurations. This is easy because the spins do not interact. The result is

\[
n_s = 2^n M^{-1} \sum_{m=0}^{M-1} e^{-2\pi i m\Delta/M} (\prod_{j=1}^n \cos(2\pi ma_j/M)).
\]

For \( \Delta = 0 \) and in the limit \( M \rightarrow \infty \) we have \( n_s = 2^n I_s \), where \( I_s = \int_0^{2\pi} \cos(a_1 \theta) \ldots \cos(a_n \theta) d\theta \). The question whether \( I_s = 0 \) or not is known to be equivalent to the (non-)existence of a solution of the number partitioning problem [4,23].

The above approach is easily generalized to handle constraints. Introducing another Hamiltonian \( H' = C - \sum_{j=1}^n \sigma_j^z \), the number of solutions \( n_s(C) \) to the constrained number partitioning problem is given by

\[
n_s(C) = \frac{1}{MK} \sum_{k=0}^{K-1} \sum_{m=0}^{M-1} \text{Tr} \, e^{-2\pi i mH'/M} e^{-2\pi ikH'}/K,
\]

(3)

where \( K = n + |C| + 1 \). Repeating the same steps as above we find that the sum over \( k \) yields zero unless \( C = \sum_{j=1}^N S_j = \sum_{a_j \in A_1} 1 - \sum_{a_j \in A_2} 1 \). The expression corresponding to (2) is

\[
n_s(C) = \frac{2^n}{MK} \sum_{k=0}^{K-1} \sum_{m=0}^{M-1} e^{-2\pi i m\Delta/M - 2\pi i k C/K} \times \prod_{j=1}^n \cos \left( \frac{2\pi ma_j}{M} + \frac{2\pi k}{K} \right),
\]

(4)

If \( n_s > 0 \) we can also find a partitioning in the following manner. Assume we already know the values of the first \( 0 < l - 1 < n \) spins. We make a guess for \( S_l \) and compute \( n_s(l) \equiv M^{-1} \sum_{m=0}^{M-1} \text{Tr} \, e^{-2\pi i mH/M} \) where the use of the symbol \( \text{tr} \) instead of \( \text{Tr} \) indicates that in calculating the trace, the values of the variables \( S_1, \ldots, S_{l-1} \) are fixed. If \( n_s(l) > 0 \) our guess for \( S_l \) was correct, if not we reverse \( S_l \). Then we increase \( l \) by one and repeat the procedure. This procedure finds a partitioning after \( n \) of these steps. Of course the same strategy applies to the constraint case.

The algorithms defined by (2) and (4) solve NPP’s and constrained NPP’s without recourse to dynamic programming. On a conventional computer they require a computation time bounded by \( nM \) (or \( nMK \) for the constrained case) and hence they, just like the dynamic programming algorithm, do not solve the (constrained) NPP in polynomial time (space) either. However, as we now show, on a QC algorithm (1) [(3)] solves the (constrained) NPP in polynomial time and space. As will be clear from the discussion below, it is sufficient to concentrate on algorithm (1).

The first step is to introduce a “number operator” \( X \) with eigenstates \( |x\rangle \), \( X|x\rangle = x|x\rangle \). We modify the Hamiltonian that governs the time-evolution of the QC as follows:

\[
H = \Delta X - \sum_{j=1}^n a_j \sigma_j^z X.
\]

(5)

By calculating the trace in the basis that diagonalizes \( \sigma_j^z \) and \( X \) (i.e. (5)), we find that

\[
\text{Tr} \, e^{-2\pi i H/M} = 2^n M \langle U_1 \ldots U_n U_x | e^{-2\pi i H/M} | U_1 \ldots U_n U_x \rangle.
\]

(6)

or equivalently

\[
n_s = 2^n \langle U_1 \ldots U_n U_x | e^{-2\pi i H/M} | U_1 \ldots U_n U_x \rangle,
\]

(7)

where \( |U_j\rangle \equiv (|1\rangle_j + |0\rangle_j)/\sqrt{2} \) is the uniform superposition of the spin up and down state of spin \( j \), and \( |U_x\rangle \equiv (|0\rangle + |1\rangle + \ldots + |M - 1\rangle)/\sqrt{M} \) is the uniform superposition of all the eigenstates of the number operator \( X \). In deriving expression (6) we made use of
Let us write the number of solutions (7) as \( n_s = 2^n \langle 0 | \Psi \rangle \) where \( | \Psi \rangle = U^{-1} e^{-i \sigma H/2^{p-1}} | 0 \rangle \) and \( U \equiv e^{-i \pi \sigma_j^y/4} e^{-i \pi \sigma_j^z/4} e^{-i \pi \sigma_j^z/4} e^{-i \pi \sigma_j^y/4} \). Our aim is to replace the projection onto the initial state \( | 0 \rangle \), a shorthand for the state with all spins up, by the measurement of some observable. This can be accomplished by adding another spin \( \kappa \), initially in the state of spin up, to the system and flip this spin if the other \( n + p \) are all up, i.e. by performing an AND operation on the \( n + p \) spins. With \( V \) denoting the unitary transformation that performs the AND operation we have in the language of qubits instead of spins

\[
| \Psi \rangle \equiv V U^{-1} e^{-i \sigma H/2^{p-1}} | 0 \rangle \otimes | 0 \rangle_\kappa \\
= V [2^{-n} n_s | 0 \rangle \otimes | 0 \rangle_\kappa + (\ldots) \otimes | 0 \rangle_\kappa] \\
= 2^{-n} n_s | 0 \rangle \otimes | 1 \rangle_\kappa + (\ldots) \otimes | 0 \rangle_\kappa \tag{10}
\]

where \( | \Psi \rangle \) is an element of the direct product of the Hilbert spaces spanned by the \( n + p \) spins and the auxiliary spin \( \kappa \). We use the abbreviation \((\ldots)\) to represent the sum of all states of the \( n + p \) spins that have at least one spin down. From (10) it immediately follows that

\[
n_s = 2^n \langle \Psi | (1 - \kappa^2)/2 | \Psi \rangle^{1/2} \]

It is well-known how to implement the AND operation on a QC [24]. In our practical implementation [26], we have chosen to use a three-bit network, the Toffoli-gate, as a building block for realizing the AND operation on the \( n + p \) qubits [24]. By adding extra work qubits the complete network requires of the order of \( \log_2 (n + p) \) steps and \( n + p \) extra qubits to perform the AND operation. Clearly this does not change the polynomial time and space character of the quantum algorithm that solves NPP’s. A block diagram of the complete quantum program is shown in Fig.1. We have implemented the QA on a 15-qubit QC and used it to solve the NPP’s \( A = \{1, 2, 4\}, A = \{1, 1, 1, 4\} \) and \( A = \{2, 2, 2, 4\} \) (these examples are included in the software distribution [26]). In the final state the expectation values of the 15-th qubit are 0.015625, 0.00390625 and 0 respectively. The corresponding number of solutions is \( n_s = 2, n_s = 1 \) and \( n_s = 0 \). Clearly the demonstration program correctly solves NPP problems, but because we use a conventional computer to emulate the QC, it does not solve NPP with the efficiency of a genuine QC.

In conclusion we have shown that a QC can solve NPP’s in polynomial time and memory space. With minor modifications to the quantum algorithm described above, a QC can solve another NP-complete problem, namely SUBSET-SUM [4,5], in polynomial time and memory space as well.

This work is partially supported by the Dutch ‘Stichting Nationale Computer Faciliteiten’ (NCF).
This value of $M$ is not optimal and can be reduced by a factor of about two. However this is irrelevant for what follows and we therefore omit the discussion of this technical point.