Highly Structured Searches with Quantum Computers

Tad Hogg
Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, California 94304

(Received 30 October 1997)

A quantum algorithm for a class of highly structured combinatorial search problems is introduced. This algorithm finds a solution in a single step, contrasting with the linear growth in the number of steps required by the best classical algorithms as the problem size increases, and the exponential growth required by classical and quantum methods that ignore the problem structure. In some cases, the algorithm can also guarantee that insoluble problems, in fact, have no solutions, unlike previously proposed quantum search algorithms. [S0031-9007(98)05575-6]

PACS numbers: 03.67.Lx, 02.70.–c, 03.65.Bz, 89.70.+c

Quantum computers [1–5] offer a new approach to combinatorial search problems [6] with their ability to operate simultaneously with many possible solutions. A quantum algorithm to factor large integers [7] much more rapidly than classical machines offers a dramatic example of this approach. While several additional algorithms have been developed [8–14], whether quantum computers can improve on heuristically guided classical methods for general searches remains an open question. These heuristics exploit the structure of the problems to greatly reduce the search cost in many cases, raising the question of whether this structure can also improve quantum searches.

This paper presents a new quantum search algorithm that is extremely effective for some highly structured search problems. These problems can also be readily solved with classical heuristics, i.e., they are relatively easy. However, the new quantum algorithm requires even fewer steps than the best classical methods, providing another example for which quantum computers can outperform classical ones.

Combinatorial search problems of the class known as *nondeterministic polynomial* (NP) have very many possible states and a procedure that quickly checks whether a given state is a solution [6]. Examples of such problems include scheduling, finding low energy states of spin glasses and proteins, and automatic theorem proving. A prototypical case is the satisfiability problem (SAT) which consists of a logical formula in n variables, $V_1, \ldots, V_n$, and the requirement to find an *assignment*, specifying a value (true or false) for each variable, that makes the formula true. There are 2^n assignments. The SAT problem has received considerable attention since it is one of the most difficult NP problems [6], in the sense that an effective method for solving SAT can be readily transformed into an effective method for any other NP problem (but not conversely).

The assignments of a SAT problem can also be viewed as bit strings with the ith bit being 0 or 1 according to whether V_i is assigned the value false or true, respectively. For bit strings r and s, let $|s|$ be the number of 1-bits in s and $r \wedge s$ the bitwise AND operation on r and s. Thus, $|r \wedge s|$ counts the number of 1-bits both assignments have in common. Let $d(r,s)$ be the Hamming distance between r and s, i.e., the number of positions at which they have different values, given by

$$d(r,s) = |r| + |s| - 2|r \wedge s|. \tag{1}$$

A logical formula can be expressed in various equivalent forms, e.g., as a conjunction of *clauses*. A clause is a disjunction of some variables, any of which may be negated. When all the clauses in a formula in this form have exactly k variables, the problem is an instance of k-SAT. A clause with k variables is false for exactly one set of values for its variables, and true for the other $2^k - 1$ choices. An example of such a clause for $k = 3$, with the third variable negated, is V_1 OR V_2 OR (NOT V_3), which is false for exactly one set of values: $\{V_1 = \text{false}, V_2 = \text{false}, V_3 = \text{true}\}$. Since the formula is a conjunction of clauses, a solution must satisfy every clause so each false clause for a given assignment is counted as a conflict for that assignment. Solutions are assignments with no conflicts.

In general, the computational cost of solving a SAT problem grows exponentially with the number of variables. However, a few simple cases can be solved without exponential cost by heuristics that use the regular structure of the problem to quickly and accurately determine search choices that lead to solutions. One example is given by 1-SAT problems. In this case, each clause eliminates one value for a single variable allowing classical algorithms to examine the variables independently, giving an overall search cost of $O(n)$.

A quantum computer can solve a 1-SAT problem in a *single* step. To see this, let m be the number of clauses in the 1-SAT formula. First, form the state $|\psi\rangle = 2^{-n/2} \sum_s |s\rangle$, an equal superposition of all 2^n assignments, and then compute

$$|\phi\rangle = |UR\psi\rangle, \tag{2}$$

where the matrices R and U are defined as follows. The matrix R is diagonal with $R_{ss} \equiv \rho(s)$ depending on the number of conflicts c in the assignment s, ranging from 0 to m,

$$\rho(s) = \rho_c \equiv \begin{cases} \sqrt{2}\cos[(2c - 1)\frac{\pi}{4}] & \text{for even } m, \\ i^c & \text{for odd } m, \end{cases} \quad (3)$$

which have unit magnitude. The matrix elements $U_{rs} = u_{d(r,s)}$ depend only on the Hamming distance between the assignments r and s, with

$$u_d \equiv \begin{cases} 2^{-\frac{n-1}{2}}\cos[(n - m + 1 - 2d)\frac{\pi}{4}] & \text{for even } m, \\ 2^{-n/2}e^{i\pi(n-m)/4}(-i)^d & \text{for odd } m. \end{cases} \quad (4)$$

To evaluate the behavior of this algorithm, consider a soluble 1-SAT problem with m distinct clauses, each of which must involve a distinct variable. Thus m variables have constrained values while the remaining $n - m$ are unconstrained, giving 2^{n-m} solutions. Equation (2) can be evaluated by using

$$\sum_{z=0}^{n}(-1)^z\binom{n}{z} = \delta_{n0}, \quad (5)$$

where $\delta_{xy} = 1$ if $x = y$ and 0 otherwise, and

$$\sum_{k=0}^{n}\binom{n}{k}\cos(x + ky) = 2^n\cos^n\left(\frac{y}{2}\right)\cos\left(x + \frac{ny}{2}\right). \quad (6)$$

Consider an assignment r with h conflicts, i.e., the number of the m constrained variables to which r assigns an incorrect value. Then from Eq. (2), $\langle r|\phi\rangle = 2^{-n/2}\sum_s U_{rs}\rho(s)$. Each assignment s in this sum can be characterized by (1) x, the number of conflicts s shares with r; (2) y, the number of conflicts of s that are not conflicts of r; and (3) z, the number of the $n - m$ unconstrained variables that have different values in r and s. In terms of these values, s has $x + y$ conflicts and $d(r,s) = (h - x) + y + z$. Counting the number of assignments with given values for x, y, and z then gives

$$\langle r|\phi\rangle = 2^{-\frac{n}{2}}\sum_{xyz}\binom{h}{x}\binom{m-h}{y}\binom{n-m}{z} \times u_{h-x+y+z}\rho_{x+y}. \quad (7)$$

Substituting the values from Eqs. (3) and (4), and making use of Eqs. (5) and (6), gives $\langle r|\phi\rangle = 2^{-(n-m)/2}\delta_{h0}$. Thus, $|\phi\rangle$ has equal amplitudes among the states with no conflicts, i.e., the solutions, and no amplitude among nonsolutions. A measurement made on this final state is guaranteed to produce a solution.

Completing the description of this algorithm requires showing how it can be performed efficiently on a quantum computer. The state $|\psi\rangle$ can be created rapidly by separately mixing each of the n-bits [10]. For $|R\psi\rangle$, note that R is a unitary diagonal matrix. Thus, $|R\psi\rangle$ can be performed efficiently [8] using a reversible version of the rapid classical procedure that counts the number of conflicts.

The matrix U can be implemented in terms of two simpler matrices, W and Γ, defined as follows. For assignments r and s,

$$W_{rs} = 2^{-n/2}(-1)^{|r\wedge s|}, \quad (8)$$

which is known as the Walsh transform, and Γ is a diagonal matrix whose elements $\Gamma_{rr} \equiv \gamma(r)$ depend only on the number of 1-bits in each assignment, i.e.,

$$\gamma(r) = \gamma_h \equiv \begin{cases} \sqrt{2}\cos[(m - 2h - 1)\frac{\pi}{4}] & \text{for even } m, \\ i^h e^{-i\pi m/4} & \text{for odd } m, \end{cases} \quad (9)$$

where $h = |r|$, ranging from 0 to n, and which have unit magnitude. The matrix W is unitary and can be implemented efficiently [8,10]. Furthermore, because the elements of Γ are readily computed by counting the number of 1-bits in the corresponding assignment, Γ is an efficiently computable unitary operation [8]. Finally, U can be implemented by the product $W\Gamma W$. To see this, let $\hat{U} \equiv W\Gamma W$. Then

$$\hat{U}_{rs} = 2^{-n}\sum_{h=0}^{n}\gamma_h S_h(r,s),$$

where

$$S_h(r,s) = \sum_{t,|t|=h}(-1)^{|r\wedge t|+|s\wedge t|},$$

with the sum over all assignments t with h 1-bits. Each 1-bit of t contributes 0, 1, or 2 to $|r\wedge t| + |s\wedge t|$ when the corresponding positions of r and s are both 0, have exactly a single 1-bit, or are both 1, respectively. Thus, $(-1)^{|r\wedge t|+|s\wedge t|}$ equals $(-1)^z$ where z is the number of 1-bits in t that are in exactly one of r and s. There are $(|r| - |r\wedge s|) + (|s| - |r\wedge s|)$ positions from which such bits of t can be selected, and by Eq. (1) this is just $d(r,s)$. Counting the number of assignments t with h 1-bits of which z are in exactly one of r and s gives $S_h(r,s) = S_{hd}^{(n)}$ where

$$S_{hd}^{(n)} \equiv \sum_z(-1)^z\binom{d}{z}\binom{n-d}{h-z},$$

with $d = d(r,s)$, so that $\hat{U}_{rs} = \hat{u}_{d(r,s)}$ with $\hat{u}_d = 2^{-n}\sum_h\gamma_h S_{hd}^{(n)}$. Equation (9) then gives $\hat{u}_d = u_d$ as defined in Eq. (4). Thus $U = W\Gamma W$, allowing U to be efficiently implemented.

This algorithm efficiently solves any 1-SAT problem. With a slight modification it also applies to maximally constrained soluble k-SAT problems for any k. These problems have the largest possible number of clauses that still allows for a solution. That is, $m = m_{\max}$ where $m_{\max} = \binom{n}{k}(2^k - 1)$ because the single solution precludes any clause that matches it. Although the variables for such problems cannot be considered independently, classical heuristics are nevertheless able to solve such problems in $O(n)$ steps. For example, local search methods start with a random assignment and, at each step, examine the number of conflicts in each neighbor (i.e., the n assignments that differ in a single value) and move to a neighbor with fewer conflicts [15,16].

Let the *good* value for each variable be the value (true or false) it is assigned in the unique solution, while the opposite value is the *bad* value. An assignment with j bad values has $\binom{n-j}{k}$ sets of k variables given the same values as the solution. Each of the remaining sets conflicts with a clause in the problem since it is maximally constrained. Thus each assignment with j bad values has

$$\binom{n}{k} - \binom{n-j}{k}$$

conflicts. For such an assignment, j neighbors have $j-1$ bad values, and the remaining $n-j$ have $j+1$ bad values. While the bad values cannot be determined without first knowing the solution, the *number* of bad values, j, can usually be determined from the number of conflicts in the neighbors. Specifically, for $j \leq n-k$, s has j neighbors with fewer conflicts and $n-j$ with more. When $j = n-k+1$, the assignment continues to have j neighbors with fewer conflicts, but the remaining $k-1$ neighbors, with an additional bad value, have the same number of conflicts. Finally, the neighbors of assignments with $n-k+1 < j \leq n$ all have the same number of conflicts. Thus, the number of conflicts in an assignment's neighbors determines the value of j, with the exception that assignments with $n-k+1 < j \leq n$ are not distinguishable. Thus, for a maximally constrained k-SAT problem,

$$c_{\text{eff}}(s) = \begin{cases} j & \text{if } j \leq n-k+1\,, \\ n-k+2 & \text{otherwise} \end{cases} \tag{10}$$

can be determined rapidly, using the same method as classical local search methods [15,16]. Thus c_{eff} is a computationally tractable approximation to the number of conflicts each assignment would have in the corresponding 1-SAT problem, i.e., the maximally constrained 1-SAT problem with the same solution as the k-SAT problem. Only assignments with more than $n-k+2$ bad values are given an incorrect value of j by this approximation. In particular, the c_{eff} is always correct for $k=2$.

These observations suggest changing Eq. (3) of the algorithm to use $\rho(s) = \rho_{c_{\text{eff}}(s)}$ from Eq. (10), and using $m=n$, as appropriate for the corresponding 1-SAT problem. To see how this approximation changes the performance, consider an assignment s with c bad values and let $\delta_c = \rho_{c_{\text{eff}}(s)} - \rho_c$, which is nonzero only for $c \geq n-k+3$. Then $|\phi\rangle = |\phi^{(1)}\rangle + |U\Delta\psi\rangle$ where $|\phi^{(1)}\rangle$ is the result for the corresponding 1-SAT problem, i.e., all amplitude in the solution, and Δ is a diagonal matrix, with elements given by δ_c. The change in the amplitude in the solution state is $\eta \equiv \langle r|U\Delta\psi\rangle$ when r is the solution. Using Eq. (7) with $h=0$ and ρ_c replaced with δ_c gives

$$\eta = 2^{-n/2}\sum_{y=0}^{n}\binom{n}{y}u_y\delta_y\,, \tag{11}$$

because $m=n$ for the corresponding 1-SAT problem. Since the ρ_c have unit magnitude, $|\delta_c| \leq 2$ for $c \geq n-k+3$, and $\delta_c = 0$ otherwise. Then using Eq. (4) and a bound on the sum of binomial coefficients [17] valid when $n-k+3 \geq n/2$,

$$|\eta| \leq 2^{-(n-1)}\binom{n}{k-3}\frac{n+1-(k-3)}{n+1-2(k-3)}\,.$$

Thus, the probability to obtain a solution is

$$P_{\text{soln}} = |1+\eta|^2 \geq 1 - 2|\eta| \sim 1 - 2^{-(n-2)}\frac{n^{k-3}}{(k-3)!}\,,$$

so this algorithm will almost surely find the solution in one search step as n increases. However, when $k > 2$, $P_{\text{soln}} < 1$ so the algorithm no longer guarantees whether a solution exists.

As a simple alternative, note that the number of conflicts grows strictly monotonically for $j \leq n-k$. In these cases j can be computed directly from the number of conflicts. This alternative avoids the overhead of examining the neighbors but with the consequence that then assignments with $n-k+1 \leq j \leq n$ are not distinguishable. Whether incurring this slight additional error is justified will depend on the cost of examining neighbors or, when $k=2$, whether it is important to be able to guarantee whether a solution exists.

In summary, for 1-SAT and maximally constrained k-SAT, this algorithm finds a solution in one step almost surely as $n \to \infty$, while classical heuristics require $O(n)$ steps. By contrast, search methods that ignore the problem structure require $O(2^n)$ steps classically, and $O(2^{n/2})$ steps on quantum computers [8]. In addition, for all 1-SAT problems and maximally constrained 2-SAT problems, the algorithm finds a solution with probability one. Thus, in these cases, failure to find a solution definitely indicates the problem is not soluble. This contrasts with previously proposed quantum algorithms that find solutions with probability less than one and, hence, cannot guarantee no solutions exist.

While search algorithm performances are often compared based on the number of search steps required, i.e., the number of sequentially examined assignments, it is also important to compare the number of more elementary computational steps required. The matrix operations and forming the initial state can be done in $O(n)$ time [8]. The time required to examine the assignments, e.g., to determine their number of conflicts or comparison to their neighbors, will be comparable for both quantum and classical algorithms since they can both make use of the same procedures. This cost will depend on the effectiveness of the data structures used to compare assignments with the constraints, typically at most $O(m)$. The n neighbors of an assignment could be examined in $O(n)$ time. Thus the cost for a *single* search step is about the same for the quantum algorithm and classical searches when neighbors are examined. However, the quantum algorithm is able to examine all assignments in superposition while a classical search examines just one, allowing the quantum algorithm to complete in just one step while the best classical methods require $O(n)$ steps.

An important issue is how quantum algorithms degrade with errors and decoherence [18,19]. While there has been progress in implementation [20–24], error control [25,26], and studies of decoherence [27], it remains to be seen how these problems affect the algorithm presented here. In particular, because the algorithm requires only a single step and simple phases, decoherence is likely to be less of a difficulty for it than algorithms requiring multiple steps to move significant amplitude to solutions [10,13].

A significant generalization would be to problems with fewer constraints. Problems with an intermediate number of constraints are the most difficult for classical heuristics [28,29] as well as quantum searches based on analogies with these classical methods [12,13]. The usefulness of Eqs. (3) and (9) depends on the regular relations among the number of conflicts in neighboring assignments. With fewer constraints, these relations vary considerably among problems. Nevertheless, Eq. (11) remains small even with some additional errors in the phase choices. In particular, for most k-SAT problems with $m \gg n^2$, $|\eta|$ continues to approach 0 as $n \to \infty$, increasing the range of problems this algorithm is very likely to solve in a single step [30]. This raises the general issue of optimally using the information that can be readily determined about assignments in combinatorial searches. Such information includes whether a state is a solution, the number of conflicts it has, and how it compares with its neighbors. Additional information is available on consistency of assignments to only some of the variables, as used with incremental searches [12]. Making fuller use of this available information may improve performance by better matching characteristics of combinatorial searches to capabilities of physically realizable devices.

[1] P. Benioff, J. Stat. Phys. **29**, 515 (1982).
[2] D. Deutsch, Proc. R. Soc. London A **400**, 97 (1985).
[3] D. P. DiVincenzo, Science **270**, 255 (1995).
[4] R. P. Feynman, Found. Phys. **16**, 507 (1986).
[5] S. Lloyd, Science **261**, 1569 (1993).
[6] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979).
[7] P. W. Shor, in *Proceedings of the 35th Symposium on Foundations of Computer Science,* edited by S. Goldwasser (IEEE, Los Alamitos, CA, 1994), pp. 124–134.
[8] M. Boyer, G. Brassard, P. Hoyer, and A. Tapp, in *Proceedings of the Workshop on Physics and Computation (PhysComp96),* edited by T. Toffoli *et al.* (New England Complex Systems Institute, Cambridge, MA, 1996), pp. 36–43.
[9] V. Cerny, Phys. Rev. A **48**, 116 (1993).
[10] L. K. Grover, Phys. Rev. Lett. **78**, 325 (1997).
[11] L. K. Grover, Technical Report No. Los Alamos quant-ph/9706005, Bell Labs (unpublished).
[12] T. Hogg, J. Artif. Intell. Res. **4**, 91 (1996), available online at http://www.jair.org/abstracts/hogg96a.html.
[13] T. Hogg, Technical Report No. Los Alamos quant-ph/9701013, Xerox (unpublished).
[14] B. M. Terhal and J. A. Smolin, Technical Report No. Los Alamos quant-ph/9705041, IBM (unpublished).
[15] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird, Artif. Intell. **58**, 161 (1992).
[16] B. Selman, H. Levesque, and D. Mitchell, in *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI92)* (AAAI Press, Menlo Park, CA, 1992), pp. 440–446.
[17] E. M. Palmer, *Graphical Evolution: An Introduction to the Theory of Random Graphs* (Wiley, New York, 1985).
[18] R. Landauer, in *Proceedings of the Drexel-4 Symposium on Quantum Nonintegrability*, edited by D. H. Feng and B.-L. Hu (International Press, Boston, 1994).
[19] W. G. Unruh, Phys. Rev. A **51**, 992 (1995).
[20] A. Barenco, D. Deutsch, and A. Ekert, Phys. Rev. Lett. **74**, 4083 (1995).
[21] J. I. Cirac and P. Zoller, Phys. Rev. Lett. **74**, 4091 (1995).
[22] D. G. Cory, A. F. Fahmy, and T. F. Havel, in *Proceedings of the Workshop on Physics and Computation (PhysComp96)* (Ref. [8]), pp. 87–91.
[23] N. Gershenfeld, I. Chuang, and S. Lloyd, in *Proceedings of the Workshop on Physics and Computation (PhysComp96)* (Ref. [8]), p. 134.
[24] T. Sleator and H. Weinfurter, Phys. Rev. Lett. **74**, 4087 (1995).
[25] A Berthiaume, D. Deutsch, and R. Jozsa, in *Proceedings of the Workshop on Physics and Computation (PhysComp94)* (IEEE, Los Alamitos, CA, 1994), pp. 60–62.
[26] P. Shor, Phys. Rev. A **52**, 2493 (1995).
[27] I. L. Chuang, R. Laflamme, P. W. Shor, and W. H. Zurek, Science **270**, 1633 (1995).
[28] S. Kirkpatrick and B. Selman, Science **264**, 1297 (1994).
[29] T. Hogg, B. A. Huberman, and C. Williams, Artif. Intell. **81**, 1 (1996).
[30] T. Hogg, Technical report, Xerox (unpublished).