Exploiting the Deep Structure of Constraint Satisfaction Problems with Quantum Computers

Tad Hogg

Xerox Palo Alto Research Center
Palo Alto, CA 94304, U.S.A.
hogg@parc.xerox.com

Abstract

The deep structure of constraint satisfaction problems explains the association of hard search instances with a phase transition in problem solubility. This structure is also the basis of a quantum search algorithm exhibiting the phase transition. In this paper, this algorithm is modified to incorporate additional problem structure. This modification is an example of a general method for including heuristics in quantum search. The new algorithm is evaluated empirically for random 3SAT, illustrating how quantum searches can benefit from using problem structure, on average.

Introduction

Quantum computers may offer dramatic performance improvements [7–9, 13, 19, 24], especially in light of recent progress in implementation [1, 4, 5, 12, 25] and error correction [23] which is likely to produce quantum computers with 10 bits or so in the next few years [5, 12].

Of most relevance to AI is the open question of whether quantum computers can solve *all* NP problems in polynomial time, although this currently seems unlikely. Even if not, they still might rapidly solve most problems, which are often much easier than worst case analyses suggest. In fact, for a wide variety of search methods, hard instances are not only rare but also concentrated near phase transitions in problem behavior [14, 18].

A recently proposed quantum algorithm also exhibits the transition [17]. While effective for some classes of random problems, it does not fully use the structure of constraint satisfaction problems (CSPs). This raises the question of how an algorithm can include additional structure. This is an example of a more general question: is there a systematic way to incorporate the knowledge used in classical heuristics within quantum searches?

As an approach to addressing this question, this paper first reviews the deep structure of CSPs, previous quantum search algorithms, and the capabilities of quantum computers. A new search algorithm using additional structure is then presented, followed by an empirical evaluation using random 3SAT. The final section describes how this method for using additional structure can also be used to add heuristics to the algorithm.

Deep Structure and Search

NP search problems have exponentially many possible states and a procedure $f(s)$ that quickly checks whether a given state s is a solution. Constraint satisfaction problems [21] are an important example. A CSP consists of n variables, $v_1, \ldots, v_n$, and the requirement to assign a value to each variable to satisfy given constraints. Searches examine various *assignments*, which give values to some of the variables. *Complete* assignments have a value for every variable. Search states can also be viewed as sets of *assumptions*, where an assumption is an assignment to a single variable, e.g., $v_1 = 0$. Sets of assumptions that violate a constraint are *nogoods*. These include the *necessary nogoods*, in which some variables are assigned multiple values [26]. The remaining sets are *goods*, i.e., consistent. These sets, and their consistency relationships, form the *deep structure* of the CSP [26].

Classically, the necessary nogoods can be avoided completely by searching only among assignments. Unfortunately, no quantum procedure can incrementally produce complete assignments from smaller ones with the variety of variable orderings needed for effective search [17]. Thus incremental quantum algorithms must use the expanded search space containing necessary nogoods.

Two existing quantum algorithms repeatedly alternate testing states for consistency with a problem-independent operation. The latter operation can be precomputed for *all* problems of a given size, and could be implemented in special purpose hardware. Both algorithms are probabilistic and incomplete: they are likely to find solutions if any exist, but cannot guarantee no solutions exist. As with classical searches, the number of consistency tests (or *checks*) required by the algorithm characterizes search cost, although the detailed cost of each step will vary somewhat among different algorithms.

One of these algorithms, ignoring all problem structure, is likely to find one of S solutions among N possibilities in $O\left(\sqrt{N/S}\right)$ checks [2, 13]. Without using additional structure, the fastest classical search is generate-and-test,

requiring $O(N/S)$ checks. Thus this quantum algorithm is a substantial improvement, and is close to the best possible for unstructured quantum searches [2]. The search cost for this quantum algorithm varies only with the number of solutions, and thus, like generate-and-test, does not exhibit the phase transition, i.e., the cost does not eventually decrease as problems become more constrained.

The second algorithm [17] uses the deep structure to build solutions incrementally, but ignores the distinction between necessary nogoods, which do not depend on the particular problem instance, and the remaining nogoods, which vary from one instance to another. Alternatively, this algorithm can be viewed as exploiting the consistency relationships among sets of assumptions, while ignoring the fact that assumptions themselves have additional structure in terms of variables and values in the CSP. Each trial requires only $O(\log N)$ checks, but gives a low probability to find a solution, thus requiring multiple trials. The number of trials required is difficult to determine theoretically since the use of problem structure introduces many dependencies among the steps. Instead, as with many classical heuristic searches, the algorithm was evaluated empirically to show a substantial improvement *on average* for classes of random problems. The algorithm also exhibits the phase transition, to date the only quantum method to do so. This is significant in demonstrating that a quantum algorithm can at least exploit structure in highly constrained problems to reduce the search cost.

Quantum Computers

Quantum mechanics raises difficult conceptual questions [10, 22], but the aspects relevant for computation are readily described. These are the rich set of states available to a quantum computer and the operations that manipulate these states. These aspects are discussed in turn in this section.

Superpositions

A classical computer can be described by a string of bits, each corresponding to a physical device in one of two distinct states. A machine with n bits has $N = 2^n$ possible states, ranging from $s_0 = 0 \ldots 0$ to $s_{N-1} = 1 \ldots 1$. In programs, the bits are usually grouped into higher level constructs, such as integers and sets. For example, a CSP search state might be an assignment such as $\{v_1 = 1, v_2 = 0\}$.

Quantum computers use physical devices whose full quantum state can be controlled. For example [8], an atom in its ground state could represent a bit set to 0, and an excited state for 1. The atom can be switched between these states, e.g., with lasers of appropriate frequencies. The atom can also be placed in a uniquely quantum mechanical *superposition* of these values, denoted as $\psi_0|0\rangle + \psi_1|1\rangle$. The notation $|\alpha\rangle$ denotes the state of the computer specified by the description α.

With n bits, we have a superposition of all 2^n classical states: $\sum_{i=0}^{N-1} \psi_i|s_i\rangle$. The coefficients or *amplitudes* ψ_i in this sum are complex numbers. At a higher level of description, $\frac{1}{2}|\{v_1 = 1, v_2 = 0\}\rangle + \frac{\sqrt{3}}{2}|\{v_1 = 1, v_2 = 1\}\rangle$ is an example of a superposition that could arise during a search. When the states $|s_i\rangle$ are understood from context, the superposition can be represented simply as a vector of amplitudes, $(\psi_1, \psi_2, \ldots)$, called the *state vector*.

The amplitudes have a physical interpretation: when the computer's state is measured [3], e.g., to determine a definite answer to a computation, the superposition changes to be the single classical state $|s_m\rangle$ with probability $|\psi_m|^2$. Because probabilities sum to one, the amplitudes satisfy the normalization condition $\sum_i |\psi_i|^2 = 1$.

Programs for Quantum Computers

Quantum computers can run classical programs, provided they are reversible, i.e., the result has enough information to uniquely reconstruct the initial state. Because quantum mechanics is linear, a program P operates independently and simultaneously on each part of a superposition, i.e., $P(\sum \psi_i|s_i\rangle) = \sum \psi_i|P(s_i)\rangle$. This *quantum parallelism* allows a machine with n bits to operate simultaneously with 2^n different classical states.

For example, suppose P extends an assignment in a backtrack style of search, e.g., by adding a value for the second variable once the first is assigned. This takes the state $|\{v_1 = 0\}\rangle$ to $|\{v_1 = 0, v_2 = 0\}\rangle$, just as a classical machine would. It also changes the superposition $\frac{1}{2}|\{v_1 = 0\}\rangle + \frac{\sqrt{3}}{2}|\{v_1 = 1\}\rangle$ to $\frac{1}{2}|\{v_1 = 0, v_2 = 0\}\rangle + \frac{\sqrt{3}}{2}|\{v_1 = 1, v_2 = 0\}\rangle$.

Additional operations change the amplitudes of superpositions. One of these is the measurement process described above. Others create and modify superpositions. Linearity and the normalization condition limit these to unitary linear operators [17], i.e., operating on $\sum_i \psi_i|s_i\rangle$ produces a new superposition $\sum_j \psi'_j|s_j\rangle$ where the new amplitudes are related to the old ones by a matrix multiplication $\psi'_j = \sum_i U_{ji}\psi_i$ and the matrix U is unitary[2].

For example, consider a single bit set to zero, $|0\rangle$, with state vector $(1, 0)$. The unitary matrix $U = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$ converts this to $(1, 1)/\sqrt{2}$, i.e., the superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Similarly, starting from $|1\rangle$ the matrix U gives $\frac{1}{\sqrt{2}}(-|0\rangle + |1\rangle)$. In either case, the probability that a subsequent measurement produces the state $|0\rangle$ is $\left(\pm 1/\sqrt{2}\right)^2 = \frac{1}{2}$. That is, starting with either value for the bit, operating with U and measuring gives $|0\rangle$ with probability $1/2$. On the other hand, starting with an equal superposition of the two classical values, $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$,

[2] A complex matrix U is unitary when $U^\dagger U = I$, where $U^\dagger$ is the transpose of U with all entries changed to their complex conjugates. Examples include permutations, rotations and multiplication by phases (complex numbers whose magnitude equals one).

operating with U gives $|1\rangle$, which has no probability to measure $|0\rangle$. The contributions to the final amplitude of $|0\rangle$ from the two states in the original superposition exactly cancel each other, an example of destructive interference. This capability for interference [10] distinguishes quantum computers from probabilistic classical machines.

A Quantum Search Algorithm

This section describes how a new algorithm is obtained from the previous structure-based algorithm [17] by explicitly distinguishing necessary nogoods. For a CSP with n variables, the algorithm consists of the following parts:

1. *initialize:* create a superposition with equal amplitude in all goods of size k
2. *check and increment:* for $i = k$ to $n - 1$
   a. adjust phases: multiply the amplitude of inconsistent sets by -1
   b. change the superposition to consist of sets of size $i+1$ using the matrix U, described below, appropriate for the parameters n and i
3. *measure* the final superposition

In part 1, the value of k is somewhat arbitrary, provided only that it is computationally feasible to find all goods of size k. If k is held constant as n increases, the number of sets of size k grows only polynomially with n giving a feasible initial state. One choice is starting with the empty set, i.e., $k = 0$. Another choice has k equal to the size of the smallest nogoods from the problem constraints. For CSPs whose constraints each involve just a few variables this is also a feasible initial state, and is the one used in the experiments reported below.

In each step in part 2, the sets in the superposition become larger by adding one new assumption in all possible ways. Thus after the last step, all sets contain n assumptions. Finally, the measurement of part 3 produces a solution with probability

$$p = \sum_s |\psi_s|^2 \qquad (1)$$

with the sum over all solutions s. On average, this algorithm must be repeated $T = 1/p$ times to get a solution.

Phase Adjustment

The phase adjustment is the only step of the algorithm that depends on the specific problem being solved, and is the same as used in prior quantum algorithms [13, 17]. To illustrate this step in more detail, let the program have a variable s equal to the current set and a one-bit variable x, initially set to zero. Thus a single state is $|s, x = 0\rangle$. On this state we use the classical test procedure $f(s)$ that returns 0 or 1 according to whether the set s is consistent or not, respectively. A reversible way to use this procedure is $x \rightarrow x$ XOR $f(s)$, where XOR is the exclusive-or: equal to 1 if and only if exactly one of its arguments is 1. This operation gives $|s, x = f(s)\rangle$. We now operate with a unitary matrix on the single bit representing the value of x: $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, while all other aspects of the state (i.e., the set s) are unchanged. This operation, which can be done rapidly [24], multiplies a state with $x = 1$ by -1, and leaves unchanged a state with $x = 0$, giving $(-1)^{f(s)}|s, x = f(s)\rangle$. Finally, to use x again in further steps of the algorithm, it must be returned to its initial value of 0. This can be done by repeating the operation[3] $x \rightarrow x$ XOR $f(s)$, giving finally $(-1)^{f(s)}|s, x = 0\rangle$. The net result of these operations is to change the sign of the state's amplitude if the set is inconsistent.

When applied to a superposition of many sets, this procedure operates independently and simultaneously on each component of the superposition. Thus the amplitudes of all inconsistent sets are multiplied by -1. This ability to operate on exponentially many states in superposition is a dramatic example of quantum parallelism.

Incrementing Sets of Assumptions

Each value of i in part 2 of the algorithm requires a matrix that maps sets of size i to those of size $i + 1$. Thus it suffices to explicitly describe only the part of the matrix involving these sets, rather than all sets of the lattice. The remainder of the matrix is unspecified except for the requirement that it is unitary. This is always possible to achieve provided the portion of the matrix acting on sets of size i has columns that are mutually orthogonal and normalized, i.e., $U^\dagger U = I$.

Motivated by breadth-first classical search, a simple choice would be to map each assignment equally to its superset assignments, giving no amplitude to necessary nogoods. However, such a matrix is not unitary. Consider instead the unitary matrix V closest to mapping each set to its supersets. The elements[4] V_{rs} depend only on the number of assumptions the sets r and s have in common, i.e., $|r \cap s|$. This matrix treats all nogoods equally, thus mapping considerable amplitude to the necessary nogoods. To reduce this difficulty, define a new matrix U whose elements depend on three properties of the sets: $|r \cap s|$ and the number of duplicate variables they each have. As with the matrix V, the values also depend on the number of variables n and the step i but are independent of the specific problem being solved. This structure for U allows the mapping to differ for assignments (with no duplicates) and necessary nogoods (which have duplicates). Among such matrices U is selected to maximize the value of U_{rs} for the case when $s \subset r$ and there are no duplicate variables in either s or r, subject to the unitarity constraint $U^\dagger U = I$. This modifies the previous

[3] The double use of $f(s)$ can be avoided by a more complex initial choice for x [2].

[4] These values are available on-line [17].

algorithm [17] to maximize the amplitude contributed to superset assignments rather than necessary nogoods.

This constrained optimization for the values of U is solved numerically using Lagrange multipliers [20] starting from $U_{rs} = V_{rs}$. In this way, values were obtained for the matrix elements used in the results presented below[5]. The matrix U grows exponentially large with i and n, but the number of distinct values for the elements and distinct equations for the unitarity conditions grows only polynomially with i and n. Thus finding values for the elements of U involves only a polynomially large number of variables and equations, and can be solved off-line once for all CSPs of a given size. Moreover, the simple structure of U, where the matrix elements depend only on a few characteristics of the sets, allows a recursive procedure to evaluate its product with state vectors as a direct generalization of the recursive evaluation of the matrix V [16].

An Example

Consider a CSP with two variables v_1 and v_2, each with two values, 0 and 1, and the single constraint that $v_1 \neq 1$. Suppose we start with the state $|\emptyset\rangle$, the empty set with no assigned variables. The step for $i = 0$ maps this to all sets of size one, giving the state vector $(1, 1, 1, 1)/2$ with the sets considered in lexicographic order:

$$\frac{1}{2}(|\{v_1{=}0\}\rangle + |\{v_1{=}1\}\rangle + |\{v_2{=}0\}\rangle + |\{v_2{=}1\}\rangle) \tag{2}$$

The $i = 1$ step first adjusts phases to give $\phi = \frac{1}{2}(1, -1, 1, 1)$ since $\{v_1 = 1\}$ is the only nogood. This superposition is then mapped to the sets of size 2:

$$\begin{aligned} &|\{v_1{=}0,v_1{=}1\}\rangle, |\{v_1{=}0,v_2{=}0\}\rangle, |\{v_1{=}0,v_2{=}1\}\rangle, \\ &|\{v_1{=}1,v_2{=}0\}\rangle, |\{v_1{=}1,v_2{=}1\}\rangle, |\{v_2{=}0,v_2{=}1\}\rangle \end{aligned} \tag{3}$$

The first and last states are necessary nogoods, while the 2nd and 3rd are the solutions. This step requires a matrix with the structure described above, i.e., of the form

$$U = \begin{pmatrix} u & u & v & v \\ x & y & x & y \\ x & y & y & x \\ y & x & x & y \\ y & x & y & x \\ v & v & u & u \end{pmatrix} \tag{4}$$

To see this, consider the first column, containing elements U_{rs} mapping from the set $s = \{v_1 = 0\}$ to the six sets of size 2 in Eq. 3. The first element in this column is for $r = \{v_1 = 0, v_1 = 1\}$ which has one duplicate variable and one assumption in common with s. This matrix element is denoted as u. The 2nd and 3rd elements in the first column are for the sets $r = \{v_1 = 0, v_2 = 0\}$ and $r = \{v_1 = 0, v_2 = 1\}$, respectively. Both of these sets have no duplicate variables and a single assumption in common with s. Hence these elements in the matrix U must have the same value, denoted by x. Similarly, the 4th and 5th elements must be the same, denoted as y, because they are both for sets with no duplicates and no assumptions in common with s. The last element in the column is for $r = \{v_2 = 0, v_2 = 1\}$, which has a single duplicate variable and nothing in common with s, giving a new value v for the matrix. Similarly, the remaining columns must use the same values, but in different orders.

The unitarity conditions for real values are

$$\begin{aligned} E_1 &\equiv 2x^2 + 2y^2 + u^2 + v^2 - 1 = 0 \\ E_2 &\equiv 2uv + x^2 + 2xy + y^2 = 0 \\ E_3 &\equiv u^2 + v^2 + 4xy = 0 \end{aligned} \tag{5}$$

The first equation normalizes each column of the matrix, and the remaining two make the columns mutually orthogonal. We maximize the value of x subject to these conditions, because this is the matrix value that maps from assignments with one variable to those with a single additional one. That is, we maximize $F = x + \sum_{j=1}^{3} \lambda_j E_j$ by setting to zero its derivatives with respect to the variables x, y, u, v and the Lagrange multipliers $\lambda_1, \lambda_2, \lambda_3$. The maximum is at $u = -v \approx 0.35$, $x \approx 0.60$ and $y \approx -0.10$.

Thus the $i = 1$ step gives $U\phi = (v, x, x, y, y, u)$. The probability of finding a solution is $2x^2 \approx 0.73$, compared to 1/3 from random selection among the six sets of size 2, and to 1/2 from random selection among complete assignments. So, while not perfect, the algorithm is able to significantly concentrate amplitude into the solutions.

Behavior of the Quantum Search

The new algorithm's average performance was evaluated with a classical simulation[6] [16] using random 3SAT. This CSP requires values (true or false, or, respectively, 1 or 0) for n variables that make a given propositional formula true. This problem has $2n$ assumptions but exponentially many assignments. In 3SAT, the formula is a conjunction of clauses, each of which is a disjunction of 3 (possibly negated) variables. An example, with the third variable negated, is $v_1 \vee v_2 \vee \overline{v_3}$, which is false for exactly one assignment for these variables: $\{v_1 = 0, v_2 = 0, v_3 = 1\}$. Thus each clause in the formula gives a single nogood of size 3, so $k = 3$ was used in the algorithm's part 1.

Soluble random 3SAT problems with c clauses were generated by first selecting c different nogoods of size 3 from among sets that are not necessary nogoods[7], and then

[5] For simplicity, the optimization considered only real values for the matrix elements.

[6] The simulation is exponentially slower than a quantum machine, limiting it to small problems. As a check on numerical precision, the normalization condition remained within about 10^{-7} of one for each trial.

[7] This differs slightly from other studies of random 3SAT in not allowing duplicate clauses in the propositional formula.

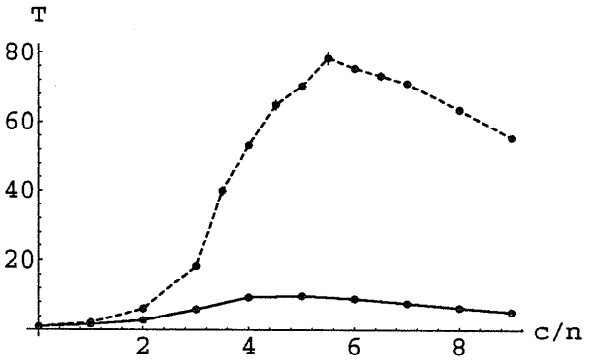

Fig. 1. Average of $T = 1/p$ vs. c/n with $n = 5$ (solid) and 10 (dashed). For each 3SAT instance, p was given by Eq. 1.

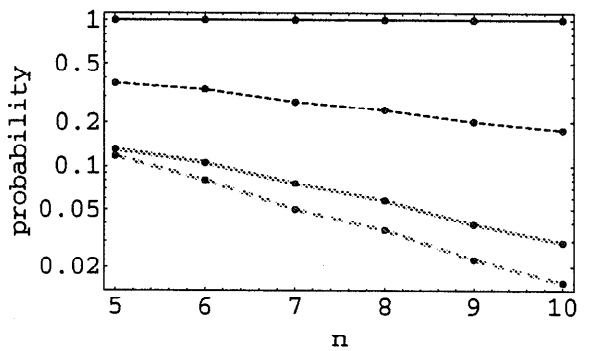

Fig. 2. Probability to find a solution, p, on a log scale, vs. n. The curves correspond to different c/n ratios: 0 (solid), 2 (dashed), 4 (gray) and 6 (gray, dashed).

discarding those problems with no solution as determined by classical backtrack. Hard problems are concentrated near the transition [6] at $c = 4.2n$. The results reported here are averages over 1000 soluble instances for each value of c and n. The figures also include error bars, showing the standard deviation in the average, which in most cases are smaller than the size of the plotted points.

Fig. 1 shows the new algorithm exhibits the phase transition. The algorithm is particularly good for underconstrained problems, in spite of the necessary nogoods. E.g., a SAT problem with $n = 10$ has $2^n = 1024$ complete assignments but $\binom{2n}{n} = 184756$ sets of size 10. At the extreme, for (trivial) problems with no constraints ($c = 0$), $p = 1$ so the new algorithm completely avoids the necessary nogoods in this case. By contrast, the original algorithm that fails to distinguish necessary nogoods from other sets [17] has p somewhat less than one for $c = 0$. This improvement also applies to underconstrained problems with $c > 0$. For highly constrained problems, the necessary nogoods are relatively less important compared to the nogoods from the problem constraints, resulting in less benefit from the modified mapping and in some cases slightly lower performance. Thus the degree of constraint could help select among search methods, as is the case for classical methods [11, 15].

The new algorithm's average scaling behavior for random 3SAT is shown in Fig. 2. This is a substantial improvement over random selection, in spite of the expanded search space. However, the small problem sizes possible with classical simulation prevent a definitive determination of whether the scaling, especially for the underconstrained problems, is polynomial rather than exponential.

Comparing the values of T with $\sqrt{N/S}$ for these problems with various values of c/n shows the new algorithm is more effective than the unstructured search algorithm [13] when the latter ignores the structure of assumptions and operates in the full search space, i.e., using $N = \binom{2n}{n}$. For example, with $n = 10$ and $c/n = 4$, $T = 53.3$ and $S = 5.4$ so $\sqrt{N/S} = 185$. At $c/n = 9$, $T = 55.6$ and $\sqrt{N/S} = 410$. The unstructured algorithm can easily be restricted to use complete assignments only, with $N = 2^n$. In this latter case, T is larger than $\sqrt{N/S}$, which is 14 and 31 for $c/n = 4$ and $c/n = 9$, respectively. From either perspective, the new algorithm is relatively better for highly constrained problems. With additional constraints the number of solutions in a soluble problem declines to $S = 1$, after which the unstructured algorithm's cost does not change while the new method's cost decreases. The unstructured algorithm using complete assignments is already near the lowest possible cost for unstructured methods [2], while the new algorithm can incorporate additional structure. Currently, the cost of the best classical method scales as $2^{n/19.5}$ at the transition [6], which is better than these quantum searches. Hence if the scaling seen here for small problems continues for larger cases, quantum searches must use additional structure if they are to improve on classical methods.

Discussion

Quantum searches can use problem structure and exhibit phase transitions. Although this requires a larger search space, structured searches incrementally concentrate amplitude toward the solutions, resulting in much better performance than random selection in the smaller space [17]. While it is less clear how they compare with the best possible unstructured quantum search [2], structured methods have the advantage of decreasing cost for highly constrained problems. The modification introduced in this paper partially ameliorates the expanded search space, especially for underconstrained problems. This modification is readily applied to reduce the amplitude mapped to necessary nogoods in other CSPs, such as graph coloring [16], with matrices appropriate for variables that have more than two values. An open question is whether other modifications would be more suitable for highly constrained problems. This could lead to new matrices appropriate for problems with different degrees of constraint. If so, a problem's constrainedness could be used to select the most

appropriate mapping, providing another example of how problem parameters can be used to improve search [11].

The exponentially slow classical simulations limit the empirical evaluation of quantum search, highlighting the need for a theoretical analysis of the structured quantum methods. Nevertheless, considering only theoretically simple algorithms (such as the unstructured search) is likely to miss many more effective possibilities. In particular, the work reported here suggests a general way to add heuristics to a quantum algorithm. If the heuristic applies to specific instances (i.e., requires testing states for consistency), the natural place to apply it is in the phase adjustment portion of the algorithm. Information applicable more generally (e.g., domain knowledge that one value is more likely than another to lead to a solution) could be used in the problem-independent mapping to emphasize contributions to states more likely to lead to solutions. If the new desired mapping is not unitary, the closest unitary mapping with a simple structure could be used instead, as shown for the modification introduced in this paper. Even if this is not successful in the worst case, it may be so on average for some difficult real-world problems. These general strategies for constructing specialized search methods will be increasingly important as quantum computers move from theoretical constructs to actual devices.

Acknowledgments

I thank J. Gilbert, S. Ganguli, J. Lamping, S. McIlraith, S. Vavasis and C. Williams for helpful discussions.

References

1. Adriano Barenco, David Deutsch, and Artur Ekert. Conditional quantum dynamics and logic gates. *Physical Review Letters*, 74:4083–4086, 1995.
2. Michel Boyer, Gilles Brassard, Peter Hoyer, and Alain Tapp. Tight bounds on quantum searching. In T. Toffoli et al., editors, *Proc. of the Workshop on Physics and Computation (PhysComp96)*, pages 36–43, Cambridge, MA, 1996. New England Complex Systems Institute.
3. V. B. Braginsky, F. Y. Khalili, and Kip S. Thorne. *Quantum Measurement*. Cambridge University Press, 1992.
4. J. I. Cirac and P. Zoller. Quantum computations with cold trapped ions. *Physical Review Letters*, 74:4091–4094, 1995.
5. David G. Cory, Amr F. Fahmy, and Timothy F. Havel. Nuclear magnetic resonance spectroscopy: An experimentally accessible paradigm for quantum computing. In T. Toffoli et al., editors, *Proc. of the Workshop on Physics and Computation (PhysComp96)*, pages 87–91, Cambridge, MA, 1996. New England Complex Systems Institute.
6. James M. Crawford and Larry D. Auton. Experimental results on the crossover point in random 3SAT. *Artificial Intelligence*, 81:31–57, 1996.
7. D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. R. Soc. London A*, 400:97–117, 1985.
8. David P. DiVincenzo. Quantum computation. *Science*, 270:255–261, 1995.
9. R. P. Feynman. Quantum mechanical computers. *Foundations of Physics*, 16:507–531, 1986.
10. Richard P. Feynman. *QED: The Strange Theory of Light and Matter*. Princeton Univ. Press, NJ, 1985.
11. Ian P. Gent, Ewan MacIntyre, Patrick Prosser, and Toby Walsh. The constrainedness of search. In *Proc. of the 13th Natl. Conf. on Artificial Intelligence (AAAI96)*, pages 246–252, Menlo Park, CA, 1996. AAAI Press.
12. Neil A. Gershenfeld and Isaac L. Chuang. Bulk spin-resonance quantum computation. *Science*, 275:350–356, 1997.
13. Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. of the 28th Annual Symposium on the Theory of Computing (STOC96)*, pages 212–219, 1996.
14. Tad Hogg. Phase transitions in constraint satisfaction search. A World Wide Web page with URL ftp://parcftp.xerox.com/pub/dynamics/constraints.html, 1994.
15. Tad Hogg. Exploiting problem structure as a search heuristic. Technical report, Xerox PARC, January 1995.
16. Tad Hogg. A framework for quantum search heuristics. In T. Toffoli et al., editors, *Proc. of the Workshop on Physics and Computation (PhysComp96)*, pages 140–146, Cambridge, MA, 1996. New England Complex Systems Institute.
17. Tad Hogg. Quantum computing and phase transitions in combinatorial search. *J. of Artificial Intelligence Research*, 4:91–128, 1996. Available online at http://www.jair.org/abstracts/hogg96a.html.
18. Tad Hogg, Bernardo A. Huberman, and Colin Williams. Phase transitions and the search problem. *Artificial Intelligence*, 81:1–15, 1996.
19. Seth Lloyd. A potentially realizable quantum computer. *Science*, 261:1569–1571, 1993.
20. D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 2nd edition, 1989.
21. Alan Mackworth. Constraint satisfaction. In S. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 285–293. Wiley, 1992.
22. N. David Mermin. Is the moon there when nobody looks? Reality and the quantum theory. *Physics Today*, pages 38–47, April 1985.
23. P. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52:2493–2496, 1995.
24. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In S. Goldwasser, editor, *Proc. of the 35th Symposium on Foundations of Computer Science*, pages 124–134. IEEE Press, November 1994.
25. Tycho Sleator and Harald Weinfurter. Realizable universal quantum logic gates. *Physical Review Letters*, 74:4087–4090, 1995.
26. Colin P. Williams and Tad Hogg. Exploiting the deep structure of constraint problems. *Artificial Intelligence*, 70:73–117, 1994.