

Graph Coloring with Quantum Heuristics

Alex Fabrikant
EECS Dept., UC Berkeley
Berkeley, CA 94720

Tad Hogg
HP Labs
Palo Alto, CA 94304

Abstract

We present a quantum computer heuristic search algorithm for graph coloring. This algorithm uses a new quantum operator, appropriate for nonbinary-valued constraint satisfaction problems, and information available in partial colorings. We evaluate the algorithm empirically with small graphs near a phase transition in search performance. It improves on two prior quantum algorithms: unstructured search and a heuristic applied to the satisfiability (SAT) encoding of graph coloring. An approximate asymptotic analysis suggests polynomial-time cost for hard graph coloring problems, on average.

Introduction

To date, quantum computers [5, 7] appear to give substantial improvement for only a few problems, most notably integer factoring [21]. At first sight, this is puzzling since quantum computers can evaluate all combinatorial search states in about the same time a conventional machine evaluates just one. Hence one might expect high performance for problems having a rapid test of whether a state is a solution, i.e., NP problems, which are the main computational bottleneck in numerous AI applications.

Unfortunately, beyond the difficulties of building quantum computers, it appears impossible to reliably and rapidly extract an answer from this simultaneous evaluation for the worst cases of NP problems [1]. Of more practical interest is the average performance of quantum algorithms that use problem structure to guide search [11, 2, 13, 16, 6]. As with conventional heuristics, such algorithms are difficult to evaluate theoretically. Moreover, empirical evaluation is also limited because, currently, quantum algorithms must be simulated on conventional machines, exponentially increasing the required memory and run time. Hence, quantum heuristics can only be tested on much smaller problems than is possible for conventional algorithms.

Despite these difficulties, insights into the structure of NP problems, particularly when formulated as constraint satisfaction problems (CSPs), help understand the capabilities of quantum computers for typical searches. One significant insight is the analogy between CSPs and physical phase transitions [4, 23, 19], which has led to new heuristics for conventional machines [12, 8, 17]. However, conventional algorithms sample only a tiny, and deliberately unrepresentative, fraction of the search states. Thus insight into

average properties can be difficult to exploit with conventional machines, particularly when the properties also have large variance, as is the case with phase transitions. On the other hand, quantum computers, by operating on the entire search space at once, can directly utilize knowledge of the average properties of search states. More broadly, quantum algorithms may motivate studies that provide new insight into typical search structure, particularly correlations among search state properties that hold on average over the whole space but not strongly enough on individual states for conventional heuristics to exploit. Gaining such insight is a fundamental concern for AI search applications, in contrast to the worst-case studies of theoretical computer science, including much of the work on quantum computation.

This paper illustrates these ideas with a quantum heuristic search algorithm for graph coloring. Graph coloring is an interesting problem for quantum algorithm design due to its structured constraints, solution invariance under permutation of colors, and nonbinary-valued variables. It thus provides additional symmetry and representational issues compared to prior studies for less structured problems such as satisfiability. Nevertheless, graph coloring is simple enough that its search space structure and phase transitions are well-understood. The operators and representations we introduce may also be useful for other, more structured, problems.

Graph Coloring

Graph coloring requires assigning one of k colors to each node in a graph so that no edge links nodes with the same color. We consider the NP-complete case of $k = 3$.

As with many search problems, graph coloring exhibits a phase transition in solubility. In our studies, we use the random graph ensemble where each graph with m edges and n nodes is equally likely to be selected. For large n the transition is near $m \sim 2.25n$, so the graphs have average degree 4.5. However, for smaller n the threshold occurs at somewhat smaller ratios of m/n . Thus, for each n we use m for which about 50% of random graphs are soluble.

Heuristics are often exponentially slower near this transition point than on either side. Generating instances near the transition gives a high concentration of hard instances for testing heuristics. While we adopt this procedure in this paper, a broader evaluation would also use hard instances found away from the transition and, more importantly, examples drawn from real-world applications. The latter include graphs with various forms of hierarchical clustering or small-world structure, which are not readily exhibited in the small graphs feasible for simulation in our studies.

Algorithm

Quantum computers operate on *superpositions* of all search states. A superposition corresponds to a *state vector*, consisting of a complex number, called an *amplitude*, for each search state. From an algorithmic perspective, a quantum computer is a device for performing some rapid operations on the state vector. These operations are matrix multiplications that have only polynomially growing cost even with exponentially many states. After a series of such operations, observing the machine (usually described as “measuring its state”) probabilistically produces a single search state, with probability equal to the square of the magnitude of amplitude associated with that state. The measurement destroys the superposition, so obtaining another state requires repeating the quantum operations from scratch. A good quantum algorithm is a series of operations on the state vector giving large amplitudes for desired states (e.g., solutions to a search problem). Measurement will then be likely to produce one of these desired states.

Our graph coloring algorithm has the same general form as unstructured search [10] and heuristic methods [13, 16] for satisfiability (SAT) and traveling salesman (TSP) problems. The quantum computer acts as a coprocessor rapidly executing an inner loop, or *trial*, of the overall algorithm, while a conventional machine examines the trial result, continuing with additional trials until a solution is found. The algorithm is incomplete, i.e., cannot determine that no solution exists, so we focus on soluble problems.

Representing Graph Coloring

The operations available with quantum computers are most naturally treated as acting on superpositions of strings of N bits (each of which is called a “qubit”). Superpositions involve the 2^N possible values for these bits. Applying this framework to graph coloring requires choosing a representation for the search states, i.e., colorings for the graph, with such a bitstring. The representation should enable efficient implementation of the operations required for the heuristic. In our case, these operations evaluate the number of conflicts in a given state and mix amplitudes of different states by multiplication with a matrix based on a distance measure between search states.

A 3-coloring of an n -node graph has several natural representations. With 3^n possible colorings, the most compact bitstring representation uses $N = \lceil \log_2 3^n \rceil$ qubits. In this representation, the distance between states is not a simple function of their bitstrings so it is unclear whether distance-based mixing can be efficiently implemented.

A second representation consists of the powerset of all possible (*variable, value*) pairs, using $3n$ qubits. This representation easily encodes the property, used for pruning in backtrack-style searches, that a conflict in a partial coloring implies a conflict in all its extensions. Our limited evaluation of this encoding shows its potential benefit does not compensate for the disadvantage of its expanded search space.

We introduce a third representation, well-suited for 3-coloring. Specifically, we associate with each node one of

four values, 0,1,2,3, where 0 indicates the node is uncolored, and the other values denote the assignment of a specific color to the node. These values need two bits per node, for a total of $N = 2n$ qubits. E.g., with four nodes, the bitstring 00 01 10 11 represents the state with node 1 uncolored, and nodes 2,3,4 assigned colors 1,2,3, respectively. While adding partial colorings expands the search space, it also allows the algorithm to use their conflict information. As detailed below, this representation readily implements mixing based on the *2-bit edit distance*, defined as the number of consecutive bit pairs (each representing a node coloring) that are distinct between two strings. For example, the distance between 00 01 10 11 and 00 10 10 01 is two, since the second and last pairs of the two bitstrings are different. This distance is a 2-bit version of Hamming distance.

In graph coloring, any permutation of a solution’s colors gives another solution. In conventional search, this symmetry can reduce the size of the search space by fixing the color choices for two nodes linked by an edge. This could also be incorporated in the representations for the quantum heuristic, but treating two nodes differently from the others complicates the analysis, so is not included in our algorithm.

Trials

Let $\psi_s^{(h)}$ be the amplitude of state s after step h of a trial. A single trial consists of the following operations:

1. initialize the amplitude equally among the states, i.e., set the amplitude $\psi_s^{(0)} = 2^{-N/2}$ for each of the 2^N states s .
2. for steps 1 through j , adjust amplitude phases based on state costs and then mix the amplitudes giving

$$\psi^{(j)} = U^{(j)} P^{(j)} \dots U^{(1)} P^{(1)} \psi^{(0)}, \quad (1)$$

where $U^{(h)}$ and $P^{(h)}$ are the mixing and phase adjustment matrices for step h , described below.

3. measure the final superposition, giving state s with probability $p(s) = |\psi_s^{(j)}|^2$.

A conventional machine then tests whether the trial’s result s is a solution and, if not, starts another trial. The probability of finding a solution with a single trial is $P_{\text{soln}} = \sum_s p(s)$ where the sum is over all solutions s . The expected number of trials to find a solution is $1/P_{\text{soln}}$, giving an expected cost of j/P_{soln} steps, each of which is comparable to a step in a conventional heuristic since it requires evaluating the conflicts in a state. This cost measure, used in the results given below, thus gives a good indication of the scaling of the cost with problem size and a relative comparison with conventional methods. More specific cost comparisons depend on the clock rates of quantum machines and how well compilers can optimize the operations, which remain open questions.

Phase Adjustment and Amplitude Mixing

The phase adjustment matrix P is diagonal and depends on the problem instance, with $P_{rr} = \exp(i\pi\rho(c(r)))$ where $c(r)$ is the cost associated with state r and $\rho(c)$ a function described below. We define a state’s cost to be the sum of

the numbers of 1) uncolored nodes and 2) edges connecting nodes of the same color. A solution is a zero-cost state.

The mixing matrix U is defined in terms of two simpler matrices: $U = WTW$. The Walsh transform W has entries

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

for states r and s , where $|r \wedge s|$ is the number of 1-bits the states have in common. The matrix T is diagonal with elements depending on the number of colored nodes in the state s . Viewed as a bitstring this is equivalent to the number of nonzero consecutive pairs $\|s\|$, e.g., $\|00011011\| = 3$. That is, $T_{ss} = \exp(i\pi\tau(\|s\|))$ with $\tau(b)$ a function described below. Quantum computers can efficiently compute the Walsh transform [2, 15], and hence U even though they involve exponentially many states. Using Eq. 2 shows the mixing matrix element U_{rs} for two states r and s has the form $u_{d(r,s)}$, i.e., depends only on the distance between the states, in analogy with conventional heuristics that examine neighbors of a given state to determine the next state to try. Because the representation allows for uncolored nodes, it is also analogous to dynamic backtracking [9] where variables can be instantiated in different orders. Unlike these conventional methods, however, U has contributions from all distances, not just neighbors. Thus the quantum operator incorporates information from the full search space in determining amplitudes for the next step.

Our 3-coloring algorithm uses the 2-bit edit distance, but generalizes to k -bit edit distance, allowing efficient implementation of mixing matrices based on distances between groups of k bits. This generalization is suitable for CSPs with more than four values per variable.

Tuning the Algorithm

Completing the algorithm requires explicit forms of $\rho(c)$ and $\tau(b)$ for each step, and a choice for the number of steps j . Ideally, these values would be selected to maximize a trial’s success for the given problem instance. Unfortunately, this is not practical since it requires detailed prior knowledge about the solutions of that instance. Instead, as with conventional heuristics, we find choices that work well on average for the ensemble of hard random graphs discussed above.

$\rho(c)$ and $\tau(b)$ can be arbitrary efficiently-computable functions. As one example, the unstructured search algorithm [10] uses $\rho(0) = 1$, $\tau(0) = 1$ and all other $\rho(c)$ and $\tau(b)$ values equal to zero. Based on a sample of small problems, functions that vary linearly with cost, distance and step give performance almost as good as allowing arbitrary functions, a property also seen with the SAT heuristic [13]. Thus, for simplicity, we restrict attention to linear functions. An overall phase factor has no effect on the probability of finding a solution, so we drop the constant terms in the linear behavior as a function of cost and distance. Thus, for step h of the algorithm, we take $\rho(c) = \rho_h c$ and $\tau(b) = \tau_h b$ with $\rho_h = \frac{1}{j}R((h-1)/j)$ and $\tau_h = \frac{1}{j}T((h-1)/j)$ where

$$\begin{aligned} R(\lambda) &= R_0 + (1-\lambda)R_1 \\ T(\lambda) &= T_0 + (1-\lambda)T_1 \end{aligned} \quad (3)$$

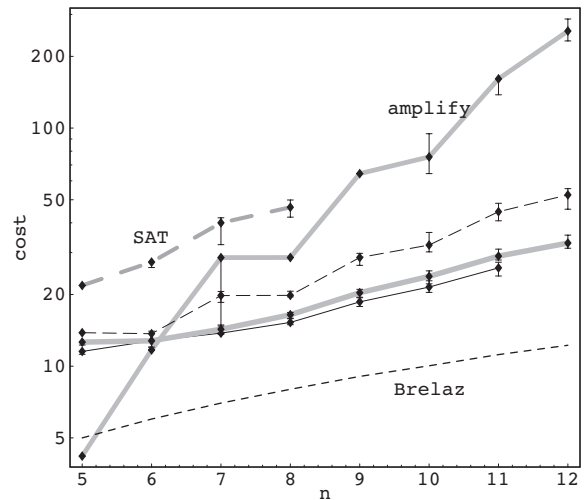


Figure 1: Median search costs vs. n . Quantum search using different phase parameter choices: optimized for each n (solid black), using $n = 6$ parameters (thick gray) and from the approximate theory (dashed black, see Section “Asymptotic Behavior”). Also shown, with labels next to the curves, are amplitude amplification on the smaller search space of the compact representation (thin gray), mapping to SAT (dashed gray, using $R_0 = 4.111$, $R_1 = -3.758$, $T_0 = 0.8288$ and $T_1 = 2.412$) and the Brelaz heuristic (dotted black), whose cost grows nearly linearly for these small problems. Graphs for $n = 5$ to 12 used 7, 10, 12, 14, 16, 18, 20 and 22 edges, respectively. Error bars show the 95% confidence intervals [22, p. 124]. For each n , the same sample of graphs was used for each method shown here.

vary linearly with the step. This choice allows explicit evaluation of the mixing matrix elements, giving

$$u_d = 2^{-2n} (1 - e^{i\pi\tau_h})^d (1 + 3e^{i\pi\tau_h})^{n-d} \quad (4)$$

for step h . We can determine good phase parameters, R_0, R_1, T_0, T_1 , from numerical optimization on a sample of graphs or by applying the theory described below.

With exponentially many steps, each trial can have $P_{\text{soln}} \approx 1$ using the unstructured algorithm [10]. Empirical evaluation for small n shows using problem structure allows better performance with a fixed number of steps, so we use $j = 10$. However, somewhat fewer steps may give better performance for the smallest sizes we consider. The approximate theory described below indicates larger problems will likely require more steps for best performance, but growing only polynomially with n .

Behavior

We numerically optimized the phase parameters for each n for a sample of 10 random soluble graphs. We then tested the heuristic with an independent sample of 200 graphs for each n . Fig. 1 compares the resulting performance to the unstructured algorithm applied to the compact representation (i.e., the first one described in Section “Representing Graph Coloring”), without assuming the number of solutions is known a priori [2]. Even though our heuristic operates in

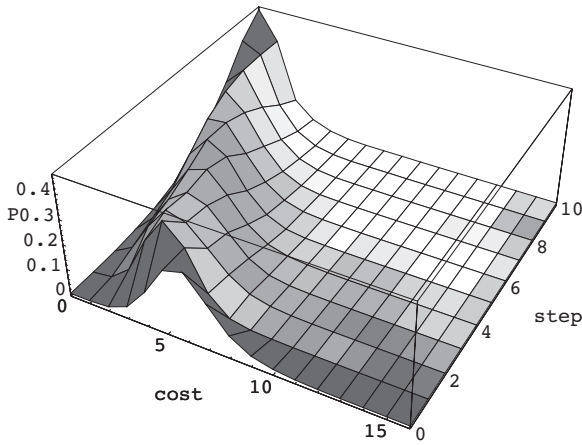


Figure 2: Amplitude shift for a graph with $n = 10, m = 18$ and 12 solutions. The plot shows the total probability in states with each cost for each step of a trial. Shading shows the relative deviation among the amplitudes for states with each cost, with lighter shadings indicating larger deviations. The deviation is relatively small for dominant cost values, i.e., near the peak in the probability for each step. This used the optimal parameters found for $n = 6$: $R_0 = 3.7032, R_1 = -2.12047, T_0 = 0.94955$ and $T_1 = 1.4052$.

a larger search space, it has lower cost and, significantly, the cost grows more slowly. The figure also shows a conventional backtrack search using the Brelaz heuristic [3]. For such small graphs, this heuristic almost always gives correct choices, avoiding any backtracking. A more significant comparison between the cost growth rates of classical and quantum methods requires larger problems to show the exponential growth of the Brelaz cost, beyond the current range of feasible quantum simulations.

Like quantum heuristics for SAT and TSP, the steps gradually shift the cost distribution toward lower costs (Fig. 2). Thus, if a trial does not yield a solution, it is still likely to yield a low-cost state, unlike unstructured search.

Numeric optimization of the phase parameters is costly, so it is useful to have a single set of parameters with reasonable performance for larger problems. Fig. 1 compares the performances using parameters optimized at each n and parameters optimized for $n = 6$ but used for all n . For all $n \neq 6$ tested, the drop in average performance is under 10%, with no visible divergence for larger problems.

SAT Mapping

Graph k -coloring has a natural mapping to k -SAT: each node maps to k variables, each denoting one color for that node, and constraints ensure each node has exactly one color and each edge connects nodes of different colors. Since quantum heuristics for k -SAT perform well on average [13], another approach to 3-coloring is mapping to 3-SAT and applying the 3-SAT heuristic. This map uses $N = 3n$, giving a larger search space than used with our graph-coloring heuristic.

The ensemble of random graphs does not map to the uniform random ensemble of SAT problems used to determine phase parameters for the quantum SAT heuristic. Thus we optimized the parameters for a sample of SAT problems created from coloring problems. Optimization became prohibitively expensive when $n > 6$. Considering the scaling results described above, we chose to compare performance of the coloring and SAT heuristics with parameters optimized at $n = 6$ for each. Fig. 1 shows the SAT heuristic lags behind the coloring heuristic. Thus our coloring heuristic uses graph coloring structure more effectively than is possible after transforming to SAT. Nevertheless, even the limited use of structure with the SAT mapping gives costs that appear to grow more slowly than unstructured search, again in spite of the larger search space.

Estimating Behavior for Large Problems

How well can quantum algorithms perform for large search problems? Ignoring problem structure gives only a quadratic improvement, far less than the improvement from exponential to polynomial cost for factoring [21]. Based on small cases, our graph coloring algorithm improves on the performance possible from unstructured algorithms but it is unclear how well it works for larger problems.

An Approximate Description

Observations with small problems indicate that states with given numbers of uncolored nodes and conflicts have similar amplitudes. This property becomes more pronounced as n increases for the states dominating the probability at each step. Among states with n_0 uncolored nodes and c conflicts, as $n \rightarrow \infty$ the overwhelming majority have nearly equal numbers of nodes with each color and about one-third of the conflicts involving each of the three colors. This motivates grouping the states according to the values $\hat{w} = (n_0, c)$ and approximating the amplitudes for such states as equal to their average value, i.e., $\psi_s^{(h)} \approx \Phi_{\hat{w}}^{(h)}$, where Φ is the average amplitude of all states with given \hat{w} .

Such “mean-field” approximations are often quite successful for large statistical systems [20]. In our case, the distribution of costs among states becomes narrower as problem size increases, leading to increasingly peaked versions of the distribution shift shown in Fig. 2. This approach also applies to a similar heuristic for random k -SAT [13].

The algorithm relies on the correlation of distance between states and their costs: nearby states tend to have many of the same conflicts and hence similar costs. We can exploit this by mixing amplitudes primarily among nearby states. Provided $j \gg 1$, Eq. 4 gives u_d equal to an irrelevant overall constant times $(-iv)^d$, with $v = \pi T / (4j)$, which decreases rapidly with d . Since the number of states at distance d from a given state is proportional to n^d , mixing is mainly among states at distances $d = O(1)$ when $j \sim n$ or larger. In particular, this means a polynomial growth in the number of steps is sufficient to ensure mixing mainly among nearby states.

With this approximation, up to an irrelevant overall normalization and phase, step h of Eq. 1 gives average amplitude of states with $\hat{W} = (N_0, C)$, namely $\Phi_{\hat{W}}^{(h)}$, in terms of

the averages for the prior step as

$$\sum_{d, \hat{w}} (-iv)^d e^{i\pi\rho(c, n_0)} \Phi_{\hat{w}}^{(h-1)} Q(\hat{W}, \hat{w}, d) \quad (5)$$

where Q is the average, over random 3-coloring problems with given m and n , of the number of states s with given \hat{w} values at distance d from a state described by \hat{W} . Note ρ and T depend on the step h .

We allow the phase adjustment to depend separately on the number of conflicts c and number of uncolored nodes n_0 instead of just their sum. Again we find a linear form is sufficient to give good performance, i.e., for step h we take $\rho(c, n_0) = \rho_h c + \sigma_h n_0$ with $\sigma_h = \frac{1}{j} S((h-1)/j)$ a new parameter not necessarily equal to ρ_h and $S(\lambda)$ has the same linear form as $R(\lambda)$ in Eq. 3 with parameters S_0, S_1 . In contrast, for the small problems described above, we treated both contributions to the cost the same way, i.e., we took $\sigma = \rho$, but the analysis described below suggests this is not the best choice for larger problems.

Q characterizes pairs of states and their conflicts for random graphs, independent of the quantum algorithm. It involves a product of binomial distributions, one for each of the 3 colors, of the number of edges linking nodes with the same colors. Evaluating Q is the key use of problem structure for analyzing the quantum algorithm, and is readily determined for random graphs. This approximation applies to any ensemble of graphs for which Q can be determined.

Asymptotic Analysis

Eq. 5 simplifies when mixing matrix elements decrease rapidly with d . In this case, those states giving the most contribution to $\psi_r^{(h)}$, with state r described by \hat{W} , have $n_0 \approx N_0$ and $c \approx C$. Hence, in Eq. 5 we can use a linearized expansion around the dominant values of N_0 and C to approximate $\Phi_{\hat{w}}^{(h-1)}$ by $\Phi_{\hat{W}}^{(h-1)} X^{c-C} Z^{n_0-N_0}$ where X and Z are complex numbers characterizing the behavior of the average amplitudes close to the dominant states at step $h-1$. The initial amplitudes, which are the same for all states, correspond to $X = Z = 1$. Small values of $|X|$ and $|Z|$ have amplitudes concentrated in states with few conflicts and few uncolored nodes, respectively. Thus good performance, i.e., $P_{\text{soln}} \approx 1$, in this approximation requires both X and Z to be small at the end of a trial, specifically $|X|, |Z| \ll 1/\sqrt{n}$.

With this linearization, Eq. 5 relates the values of X, Z at step h to those of the prior step. When j is large, the change in values from one step to the next is $O(1/j)$ so we can define smooth functions $X(\lambda), Z(\lambda)$ with $\lambda = h/j$ and the relation of values from one step to the next becomes a pair of differential equations for X and Z .

Asymptotic Behavior

For most phase parameter choices, numerical solution of the differential equations have $|X|$ and $|Z|$ nonzero for all steps, i.e., for $0 \leq \lambda \leq 1$. In such cases, the dominant states have a nonzero fraction of uncolored variables and conflicting

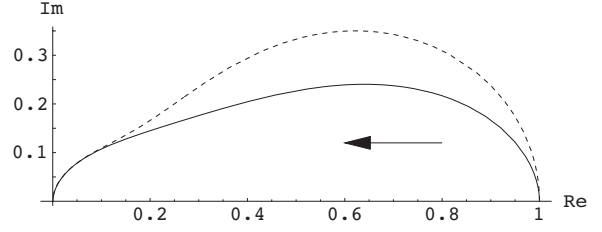


Figure 3: Behavior of $X(\lambda)$ (solid) and $Z(\lambda)$ (dashed) in the complex plane as λ ranges from 0 to 1. The initial condition is $X(0) = Z(0) = 1$ and both values approach 0 along the positive imaginary axis as $\lambda \rightarrow 1$. The parameters are $R_0 = 2.5467$, $R_1 = -2.1200$, $S_0 = 2.2521$, $S_1 = -1.1162$, $T_0 = 0.6531$ and $T_1 = 1.1759$. The arrow indicates direction of change along the curves as λ increases.

edges. However, for given μ , numerical evaluation identifies some phase parameters giving $X(1) = Z(1) = 0$. In this case, the discrete algorithm steps give final values for X, Z of size $O(1/j)$, which for $j \gg \sqrt{n}$ predicts the dominant states have the numbers of uncolored nodes and conflicts going to zero as n increases, so $P_{\text{soln}} \approx 1$ giving overall cost of $O(j)$ steps which is polynomial in n .

Fig. 3 shows the behavior for one such set of parameters. For small graphs, Fig. 1 shows the performance of the algorithm with these parameters and using the optimal number of steps for each n . That is, we examined the behavior for a few different values of j , in the range $n/2$ to n , and show the values giving the minimum median cost, i.e., j/P_{soln} . This gives $j = 4$ for $n = 5$, $j = 5$ for $n = 6, \dots, 8$ and $j = 6$ for $n = 9, \dots, 12$. Thus the best choice of j appears to grow somewhat slower than $j = O(\sqrt{n})$, a behavior also seen with solving SAT problems [14]. As one might expect, performance is worse than when using parameters optimized for small graphs. More significantly, the small sizes preclude a good evaluation of the asymptotic scaling behavior and the accuracy of the mean-field approximation.

By contrast, the unstructured search algorithm [10] gives a similar high concentration of amplitude in solutions but only after exponentially many steps. However, this result for the unstructured algorithm is exact, unlike the approximate evaluation of the heuristic method discussed here.

Discussion

Although the analysis predicts polynomial average cost, the accuracy of the mean-field approximation is an open question. Since the derivation requires $\sqrt{n} \gg 1$, empirical evaluation, limited to $n \approx 10$, does not effectively test the prediction. Nevertheless, this discussion indicates the possibility for polynomial average cost with suitable tuned parameters. The phase parameters giving $X(1) = Z(1) = 0$ according to the approximate theory are not unique and the number of steps, j , could be any power of n without precluding polynomial cost. With further analysis one could estimate the approximation error and select from among these possibilities those with the smallest error estimate. More broadly, instead of using only one choice for algorithm pa-

rameters, a portfolio of several choices gives better tradeoffs between expected performance and its variance [18].

Our mixing matrix uses the 2-bit distance, matching the structure of our problem representation. One could also try mixing matrices less directly connected to the structure. For instance, mixing based on the Hamming distance performs only slightly worse (with separately optimized parameters). Unstructured search [2] also shows this behavior. In our case, this can be understood from the heuristic’s reliance on the correlation between state costs and distance, particularly for nearby states. The 2-bit and Hamming distances are strongly correlated for nearby states, leading to similar values for the mixing matrix. This highlights the possibility of designing algorithms based on matrices matching problem structure while still being free to use other matrices for the actual implementation, provided they have similar behaviors for nearby states. This flexibility may simplify eventual hardware implementations.

Our algorithm uses quantum coherence for only one trial at a time, reducing required hardware capabilities compared to the unstructured algorithm’s need for exponentially long coherence. However, if sufficiently long coherence times become available, amplitude amplification operating on our algorithm gives a quadratic speedup to the results reported here [2]. In either case, algorithms using problem structure make significantly better use of the quantum machine.

In summary, we presented a quantum algorithm based on a generalized Hamming distance and using a simple representation including unassigned values. These features allow the algorithm to exploit more structure available in graph coloring than is possible by mapping it to satisfiability. Our approximate analysis for average behavior of large problems indicates the algorithm performs well by relying on statistical regularities throughout the search space. A definitive evaluation of this possibility for random and more structured graphs is an important open problem. More generally, this work illustrates how knowledge of problem structure from studies of CSPs can be incorporated in new quantum algorithms.

References

- [1] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh V. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26:1510–1523, 1997.
- [2] Gilles Brassard, Peter Hoyer, and Alain Tapp. Quantum counting. In K. Larsen, editor, *Proc. of 25th Intl. Colloquium on Automata, Languages, and Programming (ICALP98)*, pages 820–831, Berlin, 1998. Springer. Los Alamos preprint quant-ph/9805082.
- [3] Daniel Brelaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.
- [4] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In J. Mylopoulos and R. Reiter, editors, *Proceedings of IJCAI91*, pages 331–337, San Mateo, CA, 1991. Morgan Kaufmann.
- [5] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. R. Soc. London A*, 400:97–117, 1985.
- [6] Edward Farhi et al. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science*, 292:472–476, 2001.
- [7] Richard P. Feynman. *Feynman Lectures on Computation*. Addison-Wesley, Reading, MA, 1996.
- [8] 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.
- [9] Matthew L. Ginsberg. Dynamic backtracking. In Haym Hirsh et al., editors, *AAAI Spring Symposium on AI and NP-Hard Problems*, pages 64–70. AAAI, 1993.
- [10] Lov K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters*, 78:325–328, 1997. Los Alamos preprint quant-ph/9706033.
- [11] Lov K. Grover. Quantum search on structured problems. *Chaos, Solitons, and Fractals*, 10:1695–1705, 1999.
- [12] Tad Hogg. Exploiting problem structure as a search heuristic. *Intl. J. of Modern Physics C*, 9:13–29, 1998.
- [13] Tad Hogg. Quantum search heuristics. *Physical Review A*, 61:052311, 2000. Preprint at publish.aps.org/eprint/gateway/eplist/aps1999oct19.002.
- [14] Tad Hogg. Solving random satisfiability problems with quantum computers. Los Alamos preprint quant-ph/0104048, 2001.
- [15] Tad Hogg, Carlos Mochon, Eleanor Rieffel, and Wolfgang Polak. Tools for quantum algorithms. *Intl. J. of Modern Physics C*, 10:1347–1361, 1999. Los Alamos preprint quant-ph/9811073.
- [16] Tad Hogg and Dmitriy Portnov. Quantum optimization. *Information Sciences*, 128:181–197, 2000. Los Alamos preprint quant-ph/0006090.
- [17] Eric Horvitz et al. A Bayesian approach to tackling hard computational problems. In *Proc. of the 17th Conference on Uncertainty and Artificial Intelligence*, 2001.
- [18] Sebastian M. Maurer, Tad Hogg, and Bernardo A. Huberman. Portfolios of quantum algorithms. *Physical Review Letters*, 87:257901, 2001. Los Alamos preprint quant-ph/0105071.
- [19] Remi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. Determining computational complexity from characteristic “phase transitions”. *Nature*, 400:133–137, 1999.
- [20] Manfred Opper and David Saad, editors. *Advanced Mean Field Methods: Theory and Practice*. MIT Press, Cambridge, MA, 2001.
- [21] 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, Los Alamitos, CA, November 1994. IEEE Press.
- [22] George W. Snedecor and William G. Cochran. *Statistical Methods*. Iowa State Univ. Press, Ames, Iowa, 6th edition, 1967.
- [23] Colin P. Williams and Tad Hogg. Exploiting the deep structure of constraint problems. *Artificial Intelligence*, 70:73–117, 1994.