Applying Grover's Algorithm to Unique-k-SAT

Vasco Portilheiro

September 14, 2018

1 Introduction

Quantum computing is often lauded for its potential to greatly speed up to solutions to known problems. Although successes such as Shor's algorithm and Grover's algorithm [3] prove that quantum computing can accomplish what is classically impossible, it has proven hard to generalize or innovate new such quantum computing techniques. There is thus something to be said for exploring the application of these known quantum algorithms to classically hard problems.

In this paper, we show how Grover's algorithm can be applied to solve the Boolean unique k-satisfiability problem (Unique-k-SAT): the problem of finding the unique satisfying assignment to a formula in conjunctive normal form (CNF), where each clause contains at most k of n Boolean variables. This problem is known to be NP-complete. We find that applying Grover's algorithm solves Unique-k-SAT with the competitive asymptotic running time of $O(2^{n/2})$, which is competitive with the current best runtime for deterministic and randomized solution in the case where $k = 3$ of $O(1.307^n)$ [7], and does better than the best known classical runtime for $k = 4$ of $O(1.46981^n)$ [4].

This paper will assume basic knowledge of computational complexity, as well as some familiarity with quantum computing: the quantum circuit model and the Hadamard and CNOT gates in particular. A good introduction to these topics can be found in the seminal book of Nielsen and Chuang [5].

2 Unique-k-SAT

2.1 The problem

The general SAT problem is as follows. Given are n Boolean variables $x_1, \ldots, x_n$, and a formula

$$f(x_1, \ldots, x_n) = (a_{1,1}x_1 \vee \cdots \vee a_{1,n}x_n) \wedge \cdots \wedge (a_{m,1}x_1 \vee \cdots \vee a_{m,n}x_n), \tag{1}$$

where by $a_{i,j} \in \{-1, 0, 1\}$ we denote that in each *clause* i (i.e. the each disjunction contained in parentheses as written above) each variable x_i may appear negated, not negated, or not at all. (This is a slight abuse of notation, since if $a_{i,j} = 0$, then simply replacing $a_{i,j}x_i = 0 = false$ into the formula will not give the correct result, but it gets the idea

across.) Each $a_{i,j}x_i$ where $a_{i,j} \neq 0$ — either a variable or its negation — is called a *literal*. The formula f is thus the conjunction of m clauses, each of which are disjunctions of at most n literals, and is therefore said to be in conjunctive normal form. The SAT problem is that of determining whether there is an assignment to the variables $x_1, \ldots, x_n$ such that $f(x_1, \ldots, x_n)$ evaluates to true.

In the k-SAT problem, each clause is limited to being a disjunction of at most k literals. In this paper, we will focus on the Unique-k-SAT problem, in which we are promised that the formula in uniquely satisfiable — that is, that f is true only for one set of values of $x_1, \ldots, x_n$. The problem is this case is that of *finding* the satisfying assignment.

2.2 Classical solutions

The naive solution to the Unique-k-SAT problem would be to simply iterate through all possible variable assignments until the satisfying assignment is found. As each variable in an assignment may be either true or false, there are 2^n such assignments, and this algorithm will thus take $O(2^n)$ time.

More advanced solutions are beyond the scope of this paper, although we will stop here to mention that the seminal 1998 result of Paturi, Pudlak, Saks, and Zane [6] has been improved upon recently to produce a deterministic solution to the Unique-3-SAT solution that runs in $O(1.307^n)$ time [7], as well as a probabilistic solution to the 4-SAT that runs in $O(1.46981^n)$ time [4].

3 Grover's Algorithm

Grover's algorithm was introduced in 1996 by Grover, who published it as "A fast quantum mechanical algorithm for database search" [3]. We will present a brief overview of the algorithm here, as well as a note on an often overlooked challenge one must solve to efficiently apply the algorithm in practice.

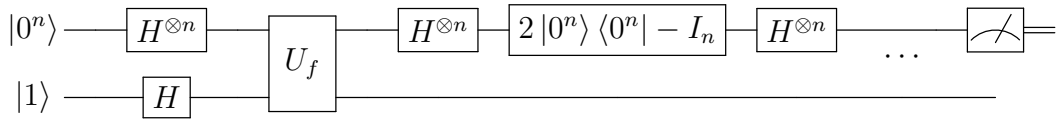
3.1 The problem Grover's solves

Although Grover himself described his algorithm as a kind of quantum database search, this view may obfuscate some aspects of what the algorithm actually does. In particular, given a function evaluatable on a quantum computer $f : X \to Y$ where $|X| = N$, and a $y \in Y$, Grover's algorithm finds with high probability $x^* \in X$ such that $f(x^*) = y$ in runtime $O(N^{1/2})$.

What does it mean for f to be "evaluatable on a quantum computer?" Traditionally, for the purposes of Grover's algorithm we consider f to be a black-box, whose function is unknown. In the quantum circuit model, we are given a control gate U_f implementing $|x\rangle |y\rangle \mapsto |x\rangle |y \oplus \mathbf{1}\{f(x) = y\}\rangle$. This allows us to evaluate f on x by using an ancilla bit (an "extra" qubit initialized to $|0\rangle$), by evaluating $U_f |x\rangle |0\rangle = |x\rangle |\mathbf{1}\{f(x) = y\}\rangle$.

3.2 The algorithm

This section's purpose is to provide a description of the algorithm, and sketch how it is that Grover's algorithm achieves the desired result under some guarantees. The circuit for Grover's algorithm is as follows.



Note that here, $n = \log_2 N$, such that the number of qubits on the first line is sufficient to represent all N elements of X with unique binary indices. The combination of gates on the top line following the U_f gate is called the *Grover diffusion operator*, for reasons that will become clear. The combination of the U_f gate and the diffusion operator is repeated $\frac{\pi N^{1/2}}{4}$ times, which is why the algorithm runs in $O(N^{1/2})$ time.

The purpose of the first $H^{\otimes n}$ gate on top line is to put the inputs to U_f in the uniform superposition

$$H^{\otimes n}\,|0^n\rangle = \frac{1}{\sqrt{N}} \sum_{x \in X} |x\rangle\,. \tag{2}$$

The purpose of the Hadamard gate on the second line is more subtle. Recall that

$$H\,|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle\,. \tag{3}$$

Thus, supposing that instead of a superposition the first line into U_f were just some $|x\rangle$, the output of U_f would be

$$\begin{aligned} U_f\,|x\rangle\,|-\rangle &= \frac{1}{\sqrt{2}} U_f\,|x\rangle\,|0\rangle - \frac{1}{\sqrt{2}} U_f\,|x\rangle\,|1\rangle && (4)\\ &= \frac{1}{\sqrt{2}} U_f\,|x\rangle\,|f(x)\rangle - \frac{1}{\sqrt{2}} U_f\,|x\rangle\,|1 \oplus \mathbf{1}\{f(x) = y\}\rangle && (5)\\ &= \begin{cases} \frac{1}{\sqrt{2}}\,|x\rangle\,|1\rangle - \frac{1}{\sqrt{2}}\,|x\rangle\,|0\rangle = -\,|x\rangle\,|-\rangle & \text{if } f(x) = y \\ \frac{1}{\sqrt{2}}\,|x\rangle\,|0\rangle - \frac{1}{\sqrt{2}}\,|x\rangle\,|1\rangle = |x\rangle\,|-\rangle & \text{if } f(x) \neq y. \end{cases} && (6) \end{aligned}$$

This is, U_f inverts the amplitude of the state x such that $f(x) = y$. Note that since we are guaranteed that the second input to U_f is $|-\rangle$, we may consider U_f restricted to the first qubit line, in which case we can write it as the operator

$$U_f = I - 2\,|x^*\rangle\langle x^*|\,, \tag{7}$$

recalling that x^* is the element of X such that $f(x^*) = y$. Since the input to U_f is the uniform superposition in (2), rather than just one state $|x\rangle$, the actual state of the system after U_f is a superposition where the probability of any $x \in X$ is the same, but where the

amplitude of the x such that $f(x) = y$ is negative, which we see by plugging in the state of the first qubit line from (2):

$$U_f \frac{1}{\sqrt{N}} \sum_{x \in X} |x\rangle = \frac{1}{\sqrt{N}} \sum_{x \in X} (I - 2\,|x^*\rangle\langle x^*|)\,|x\rangle \tag{8}$$

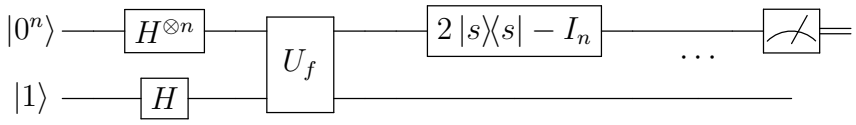
$$= -\frac{2}{\sqrt{N}}\,|x^*\rangle + \frac{1}{\sqrt{N}} \sum_{x \in X} |x\rangle \tag{9}$$

$$= -\frac{1}{\sqrt{N}}\,|x^*\rangle + \frac{1}{\sqrt{N}} \sum_{x \neq x^*} |x\rangle\,. \tag{10}$$

In order to understand the Grover diffusion operator, we will name the uniform superposition over all states in (2) to be $|s\rangle$

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x \in X} |x\rangle\,, \tag{11}$$

noting that then the circuit above can be rewritten as below.



This follows from the simple calculation that the diffusion operator can be written

$$H^{\otimes n}(2\,|0^n\rangle\langle 0^n| - I)H^{\otimes n} = 2H^{\otimes n}\,|0^n\rangle\langle 0^n|\,H^{\otimes n} - H^{\otimes n} I H^{\otimes n} = 2\,|s\rangle\langle s| - I. \tag{12}$$

Applying the diffusion operator to the output of U_f gives us the result of one iteration of Grover's algorithm:

$$(2\,|s\rangle\langle s| - I)\left(\frac{-1}{\sqrt{N}} + \frac{1}{\sqrt{N}} \sum_{x \neq x^*} |x\rangle\right) = (2\,|s\rangle\langle s| - I)(|s\rangle - \frac{2}{\sqrt{N}}\,|x^*\rangle) \tag{13}$$

$$= 2\,|s\rangle - |s\rangle - \frac{4}{N}\,|s\rangle + \frac{\sqrt{2}}{N}\,|x^*\rangle \tag{14}$$

$$= \frac{N-4}{N}\,|s\rangle + \frac{\sqrt{2}}{N}\,|x^*\rangle\,, \tag{15}$$

where we note that $\langle s|x^*\rangle = \frac{1}{\sqrt{N}}$. Thus, the probability of obtaining $|x^*\rangle$ on measurement of the first line increases with an iteration of Grover's algorithm. While we will not prove it here — as it is not the focus of this paper — this probability increases until the error probability is $O(\frac{1}{N})$ in $\frac{\pi N^{1/2}}{4}$ such iterations.

3.3 Oracle construction

One often overlooked detail required to use Grover's algorithm in practice is the implementation of the oracle U_f, as pedagogic literature often leaves it at "black box," a fact which

leads [9] to criticize the eagerness to claim that Grover's and similar algorithms efficiently solve hard problems. This concern is not misplaced. Imagine it were hard, in a strict sense, to implement the oracle. Perhaps constructing the oracle "gate" from some given function requires checking how the function operates on each possible input. Then constructing the oracle is as hard as solving the problem, and Grover's algorithm gets us nowhere. In practice, techniques exist for implementing oracles given a classical circuit implementing the oracle, which we will apply to constructing the oracle for our problem.

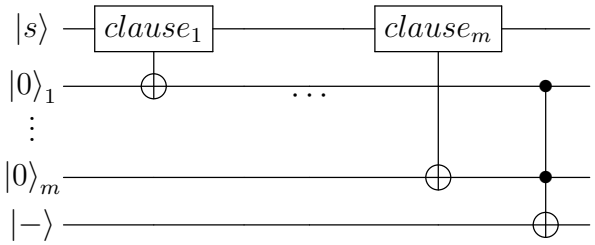
4 Applying Grover's Algorithm

4.1 Representing the problem

The representation of k-SAT for Grover's algorithm is intuitive: for each Boolean variable $x_1, \ldots, x_n$, we have one qubit. In fact, initializing these each of qubits to $|0\rangle$, we get the these Boolean variables are represented exactly by the first qubit line in the circuit diagram above, initialized to $|0^n\rangle$. With a correct implementation of the oracle, measuring this line gives a assignment of 0 or 1 to each qubit, that is to each Boolean variable, which is with high probability the satisfying assignment.

4.2 Constructing the oracle

To apply Grover's algorithm, it remains to show how to construct the oracle gate U_f from the CNF formula f. Recall that being in CNF, f is a conjunction of m clauses, where each clause is a disjunction of at most k literals. It is this natural to have an ancilla qubit for each clause, m total "clause bits." We will construct a series of NOT and CNOT gates (how exactly we will discuss shortly), which, supposing our input to U_f had only qubits in the pure states $|0\rangle$ or $|1\rangle$, would set each such ancilla qubit to be 1 if its respective clause is satisfied by the given inputs. Since f is the conjunction of these clauses, we will simply CNOT to the output ancilla qubit — the second qubit line in the circuit above — as controlled by our clause qubits. This gives us the following circuit for U_f so far, where $|s\rangle$ and $|-\rangle$ correspond to the first and second qubit lines as prepared before the first application of U_f in the circuit diagram above.



It thus remains to show how to set the m clause qubits. Note that we will also have to "uncompute" the clause bits and the input $|s\rangle$, i.e. return them to their input states $|0^m\rangle$ and $|s\rangle$, so as to correctly (and reversibly) implement U_f. This, however, will not be hard! Turning our attention to the clauses, we recall that each clause c_i is a disjunction of at $p \leq k$

literals

$$c_i = l_1 \vee \cdots \vee l_p. \tag{16}$$

However, disjunctions are not obviously amenable to quantum computation, as they are not reversible. To overcome this, we recall that by De Morgan's Law

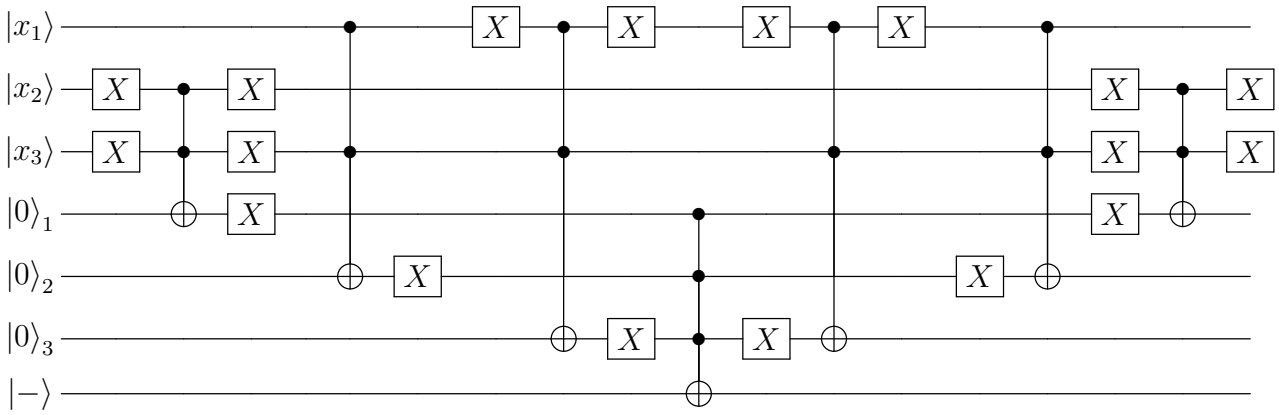
$$c_i = \neg(\neg l_1 \wedge \cdots \wedge \neg l_p). \tag{17}$$

Thus, the clause's truth value is just the negation of the CNOT gate applied to the literals' negations. Thus, we proceed as follows. Since each literal is either a variable or its negation, we simply set each qubits corresponding to variables in the clause to be negated if the literal in negated in the clause. We then negate all these "literal-valued qubits," use these as controls for CNOT with the target on the correct clause qubit, then negate the result at the clause qubit. Finally to uncompute, we perform the inverse of CNOT for each clause, which is just the same CNOT, and then we negate once more all the qubits corresponding to variables in the clause, then negate again each qubit whose literal was the negation of its variable. Note that we can make the optimization of simply not negating the each qubit whose variable's literal is the negation of that variable.

The above procedure is certainly convoluted, but in practice it is a simple pattern, and so is best explained by example. Suppose our formula were:

$$f = (x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_3). \tag{18}$$

Then following the procedure above, our U_f gate would have the following circuit.



Note first how this circuit is symmetric about the point where we CNOT onto the $|-\rangle$ "output qubit." The circuit after this point is clearly just the uncomputation, setting the input to its initial state (but, recall that in the full Grover's circuit, the amplitude of will be x^* negated). We now explain the gates before this point of symmetry. Since in the first clause x_2 and x_3 are not negated, by (17) we must negate then before using the CNOT to set the clause bit for clause 1. Also per (17), we negate this clause bit after the CNOT. In clause 2, both x_1 and x_2 are negated, so we do not need to negate their qubit lines. This is since the negation of their literals in the clause (the literals being themselves the negations of the variables) are just the variables. We then apply the CNOT and negate the result at

the clause 2 qubit. For clause 3, we only need to negate the qubit line for x_1, since x_3 is already negated in the clause.

Thus using the procedure outlined above, we have shown how to construct the correct U_f gate. The astute reader, however, will notice that clauses of more then two literals with require CNOT gates controlled by more than two qubits. While these are not part of the "standard repertoire" of quantum gates, and constructing them explicitly is beyond the scope of this paper, it is in fact possible to construct a k-controlled-NOT gate with only $k-1$ additional ancillae [8]. Thus, the for a formula f of m clauses, the oracle U_f can be constructed using only $m+k-1$ additional ancillae. Thus, our application of Grover's algorithm is efficient not only in runtime, but in space, with space usage being polynomial in the formula size.

4.3 Quantum speedup

As stated previously, the runtime of our Grover's algorithm solution to Unique-k-SAT runs time $O(2^{n/2}) \approx O(1.414^n)$. This is comparable to the $O(1.307^n)$ runtime for solution to Unique-3-SAT [7], and improves upon the best known runtime of classical algorithm for 4-SAT of $O(1.46981^n)$ [4].

5 Conclusion

The technique outlined here is by no means a novel idea, although this paper did take peculiar care in construction of the oracle. Other, more sophisticated applications of Grover's algorithm and similar techniques to SAT problems include [2] and [1]. Since finding a satisfying assignment is the same as showing that f is satisfiable, it would be interesting to extend the result above to that of checking whether a formula is satisfiable, as in general k-SAT. We expect that this should be natural, and that only the analysis would become slightly more complex, as Grover's algorithm can be extended to cases in which more than one solution exist.

6 Acknowledgements

I would like to thank Leonardo P. G. de Assis for his class in Quantum Computation, which has helped me delve deeper into my interest in quantum computing, and has given me the opportunity to apply novel ways of thinking to problem solving.

References

[1] Sheng-Tzong Cheng and Ming-Hung Tao. "Quantum cooperative search algorithm for 3-SAT". In: *Journal of Computer and System Sciences* 73.1 (2007), pp. 123–136. ISSN: 0022-0000. DOI: https://doi.org/10.1016/j.jcss.2006.09.003. URL: http://www.sciencedirect.com/science/article/pii/S0022000006001012.

[2] Evgeny Dantsin, Vladik Kreinovich, and Alexander Wolpert. "On Quantum Versions of Record-breaking Algorithms for SAT". In: *SIGACT News* 36.4 (Dec. 2005), pp. 103–108. ISSN: 0163-5700. DOI: 10.1145/1107523.1107524. URL: http://doi.acm.org/10.1145/1107523.1107524.

[3] L. K. Grover. "A fast quantum mechanical algorithm for database search". In: *eprint arXiv:quant-ph/9605043* (May 1996). eprint: quant-ph/9605043.

[4] T. Hertli. "3-SAT Faster and Simpler - Unique-SAT Bounds for PPSZ Hold in General". In: *ArXiv e-prints* (Mar. 2011). arXiv: 1103.2165 [cs.CC].

[5] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th. New York, NY, USA: Cambridge University Press, 2011. ISBN: 1107002176, 9781107002173.

[6] Ramamohan Paturi et al. "An Improved Exponential-time Algorithm for k-SAT". In: *J. ACM* 52.3 (May 2005), pp. 337–364. ISSN: 0004-5411. DOI: 10.1145/1066100.1066101. URL: http://doi.acm.org/10.1145/1066100.1066101.

[7] Daniel Rolf. "Derandomization of PPSZ for Unique-k-SAT". In: *Theory and Applications of Satisfiability Testing*. Ed. by Fahiem Bacchus and Toby Walsh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 216–225. ISBN: 978-3-540-31679-4.

[8] Siddhartha Sinha and Peter Russer. "Quantum computing algorithm for electromagnetic field simulation". In: 9 (June 2010), pp. 385–404.

[9] G. F. Viamontes, I. L. Markov, and J. P. Hayes. "Is Quantum Search Practical?" In: *eprint arXiv:quant-ph/0405001* (Apr. 2004). eprint: quant-ph/0405001.