8-1-2005

On Quantum Versions of Record-Breaking Algorithms for SAT

Evgeny Dantsin

Vladik Kreinovich

University of Texas at El Paso, vladik@utep.edu

Alexander Wolpert

Follow this and additional works at: http://digitalcommons.utep.edu/cs_techrep

Part of the Computer Engineering Commons

Comments:


Recommended Citation

http://digitalcommons.utep.edu/cs_techrep/257

This Article is brought to you for free and open access by the Department of Computer Science at DigitalCommons@UTEP. It has been accepted for inclusion in Departmental Technical Reports (CS) by an authorized administrator of DigitalCommons@UTEP. For more information, please contact lweber@utep.edu.
On Quantum Versions of Record-Breaking Algorithms for SAT

Evgeny Dantsin\textsuperscript{1}, Vladik Kreinovich\textsuperscript{2}, and Alexander Wolpert\textsuperscript{1}\textsuperscript{*}

\textsuperscript{1}Department of Computer Science, Roosevelt University
Chicago, IL 60605, USA, \{edantsin,awolpert\}@roosevelt.edu
\textsuperscript{2}Department of Computer Science, University of Texas at El Paso
El Paso, TX 79968, USA, vladik@utep.edu

Abstract

It is well known that a straightforward application of Grover’s quantum search algorithm enables to solve SAT in $O(2^{n/2})$ steps. Ambainis (SIGACT News, 2004) observed that it is possible to use Grover’s technique to similarly speed up a sophisticated algorithm for solving 3-SAT. In this note, we show that a similar speed up can be obtained for all major record-breaking algorithms for satisfiability. We also show that if we use Grover’s technique only, then we cannot do better than quadratic speed up.

1 Quantum Computing and Satisfiability

Faster quantum algorithms for SAT. In the satisfiability problem (SAT), we are given a Boolean formula $F$ in conjunctive normal form $C_1 \& \ldots \& C_m$, where each clause $C_j$ is a disjunction $l_1 \vee \ldots \vee l_k$ of literals, i.e., variables or their negations. We need to find a truth assignment $x_1 = a_1, \ldots, x_n = a_n$ that makes $F$ true. A simple exhaustive search can solve this problem in time $\sim 2^n$, where $\sim$ means equality modulo a term which is polynomial in the length of the input formula.

The main attraction of quantum computing is that it can speed up computations. In particular, Grover’s quantum algorithm [9, 10, 11, 15] searches an unsorted list of $N$ elements to find an element with a given property. In non-quantum computations, every such search algorithm requires, in the worst case, $N$ steps; Grover’s algorithm can find this element in time $O(\sqrt{N})$ with arbitrary high probability of success. Thus, a straightforward application of Grover’s technique can solve SAT in time $\sim 2^{n/2}$.

Computer simulation of quantum computing suggests that it may be possible to solve SAT even faster [12]. Can we actually use quantum computing to solve SAT faster than in time $\sim 2^{n/2}$? In this note, we discuss some aspects of this question.

\textsuperscript{*}©E. Dantsin, V. Kreinovich, and A. Wolpert, 2005
Remark.  We only consider quantum computing within the standard quantum physics. It is known that if we consider non-standard versions of quantum physics (e.g., a version in which it is possible to distinguish between a superposition of $|0\rangle$ and $|1\rangle$ and a pure state) then, in principle, we can solve NP-complete problems in polynomial time; see, e.g., [1] and references therein, and also [2, 14, 16].

Ambainis’ observation.  In [3], Ambainis considers algorithms for $k$-SAT, a restricted version of SAT where each clause has at most $k$ literals. He shows that one of the fastest algorithms for $k$-SAT, namely, the algorithm proposed by Schöning [21], can be similarly sped up from time $T \sim (2 - 2/k)^n$ to $\sqrt{T} \sim (2 - 2/k)^{n/2}$.

Schöning’s algorithm is a multi-start random walk algorithm that repeats the polynomial-time random walk procedure $S$ exponentially many times. This procedure $S$ takes an input formula $F$ and does the following:

- Choose an initial assignment $a$ uniformly at random.
- Repeat $3n$ times:
  - If $F$ is satisfied by the assignment $a$, then return $a$ and halt.
  - Otherwise, pick any clause $C_j$ in $F$ such that $C_j$ is falsified by $a$; choose a literal $l_s$ in $C_j$ uniformly at random; modify $a$ by flipping the value of the variable $x_i$ from the literal $l_s$.

As shown in [21], if the formula $F$ is satisfiable, then each random walk of length $3n$ finds a satisfying assignment with the probability $\geq (2 - 2/k)^{-n}$. Therefore, for any constant probability of success, after $O((2 - 2/k)^n)$ runs of the random walk procedure $S$, we get a satisfying assignment with the required probability. Since $S$ is a polynomial time procedure, the overall running time of this algorithm is also $T \sim (2 - 2/k)^n$. This upper bound is close to the best known upper bound for $k$-SAT (see below). Schöning’s algorithm was derandomized in [6].

In Schöning’s algorithm, there are $N \sim (2 - 2/k)^n$ results of different runs of $S$, and we look for a result in which the input formula $F$ is satisfied. Grover’s algorithm enables us to find this result in time $\sim \sqrt{N}$. More exactly, this reduction comes from the modification of the original Grover’s algorithm called amplitude amplification) [3, 5]. Thus, there exists a quantum algorithm that solves $k$-SAT in time $\sim \sqrt{T} \sim (2 - 2/k)^{n/2}$.

For 3-SAT, Schöning’s algorithm was improved by Rolf [19] to $T \sim 1.330^n$. This improvement also consists of exponentially many runs of a polynomial-time algorithm. Therefore, Rolf’s non-quantum running time $T \sim 1.330^n$ leads to the corresponding quantum time $\sqrt{T} \sim 1.154^n$.

SAT is a particular case of a more general discrete constraint satisfaction problem (CSP), where variables $x_1, \ldots, x_n$ can take $d \geq 2$ possible values, and constraints can be more general than clauses. In particular, we can consider $k$-CSP, in which every constraint contains $\leq k$ variables. Schöning’s algorithm can be naturally extended to $k$-CSP [21]. The running time of the corresponding algorithm is $T \sim (d \cdot (1 - 1/k) + \varepsilon)^n$, where $\varepsilon$ can be arbitrarily small. Similar to Schöning’s algorithm for $k$-SAT, this extension to $k$-CSP can be quantized with
the running time $T_Q \sim \sqrt{T} \sim (d \cdot (1 - 1/k) + \epsilon)^{n/2}$. A different quantum algorithm for 2-CSP is described in [4].

**The fastest algorithm for $k$-SAT.** The best known upper bound for $k$-SAT is given by the algorithm proposed by Paturi, Pudlák, Saks, and Zane [17, 18]; this algorithm is called PPSZ. This algorithm consists of exponentially many runs of a polynomial-time procedure. This procedure is based on the following approach:

- Pick a random permutation $\pi(1), \pi(2), \ldots, \pi(n)$ of the variables.
- Select a truth value of the variable $x_{\pi(1)}$ at random.
- Simplify the input formula as follows:
  - Substitute the selected truth value for $x_{\pi(1)}$.
  - If one of the clauses reduces to a single literal, simplify the formula again by using this literal.
  - Repeat such simplification while possible.
- Select a truth value of the first unassigned variable (in the order $\pi(1), \pi(2), \ldots$) at random.
- Simplify the formula as above.
- Continue this process until all $n$ variables are assigned.

As shown in [18], the PPSZ algorithm runs in time $T \sim 2^n(1 - \mu_k/k)$, where $\mu_k \to \pi^2/6$ as $k$ increases. The PPSZ algorithm was derandomized in [20] for the case when there is at most one satisfying assignment.

Since the PPSZ algorithm also consists of exponentially many runs of a polynomial-time procedure, we can use Grover’s technique to design its quantum version which requires time $T_Q \sim \sqrt{T}$.

A combination of the PPSZ and Shöning’s approaches leads to the best known upper bound for 3-SAT: $T \sim 1.324^n$ (Iwama and Tamaki [13]). Similarly to the previous algorithms, this algorithm also consists of independent runs of a polynomial-time procedure. So, by applying Grover’s algorithm, we can similarly get a quantum algorithm with time $\sqrt{T} \sim 1.151^n$.

**The fastest algorithm for SAT with no restriction on clause length.** The best known upper bound for SAT with no restriction on clause length is given in [8]. The corresponding algorithm is based on the clause shortening approach proposed by Schuler in [22]. This approach suggests exponentially many runs of the following polynomial-time procedure $S$:

- Convert the input formula $F$ to an auxiliary $k$-CNF formula $F'$. Namely, for each clause $C_j$ longer than $k$, keep the first $k$ literals and delete the other literals in $C_j$.  

*ACM SIGACT News* 3 September 2003 Vol. 34, No. 3
• Use a \(k\)-SAT algorithm, e.g., one random walk of Schöning’s algorithm, to test satisfiability of \(F'\). Assuming that \(F\) has a satisfying assignment \(a\), there are two possible cases:
  
  – First, the \(k\)-SAT algorithm has found \(a\); then we are done.
  
  – Second, some clause \(C'_j\) in \(F'\) is false under \(a\). If we guess this clause, we can reduce the number of variables in \(F\) by substituting the corresponding truth values for the variables of \(C'_j\). Therefore, we choose a clause in \(F'\) at random and simplify \(F\) by replacing the variables that occur in this clause with the corresponding truth values.

• Finally, we recursively apply \(S\) to the result of simplification.

The procedure \(S\) runs in polynomial time and finds a satisfying assignment (if any) with probability at least

\[
2^{-n \left(1 - \ln \left(\frac{m}{n}\right) + O(\ln \ln (m))\right)}.
\]

This probability can be increased to a constant by repetition in the usual way, so the algorithm for SAT requires time

\[
T \sim 2^{-n \left(1 - \ln \left(\frac{m}{n}\right) + O(\ln \ln (m))\right)}.
\]

By using Grover’s technique, we can produce a quantum version of this algorithm that requires time \(T_Q\):

\[
T_Q \sim \sqrt{T} \sim 2^{-n/2 \cdot \left(1 - \ln \left(\frac{m}{n}\right) + O(\ln \ln (m))\right)}.
\]

### 2 How Much More Can Grover’s Algorithm Help?

**At most quadratic speed-up.** So far, we have used Grover’s technique to speed up the non-quantum computation time \(T\) to the quantum computation time \(T_Q \sim \sqrt{T}\). Let us show that if Grover’s technique is the only quantum technique that we use, then we cannot get a further time reduction. Informally speaking, let us call a quantum algorithm that uses only Grover’s technique (and no other quantum ideas) *Grover-based*. We show that the following two statements hold:

• **Statement 1.** If we have a Grover-based quantum algorithm \(A_Q\) that solves a problem in time \(T_Q\), then we can “dequantize” it into a non-quantum algorithm \(A\) that requires time \(T = O(T_Q^2)\).

• **Statement 2.** If we have a non-quantum algorithm that solves a problem in time \(T\), then any Grover-based quantum algorithm for solving this problem requires time at least \(T_Q = \Omega(\sqrt{T})\).
**First statement.** Without loss of generality, we can assume that the time is measured in number of steps. Then $T_Q = t_0 + t_1 + \ldots + t_s$, where $t_0$ denotes the number of non-quantum steps in $A_Q$, $s$ denotes the number of Grover’s searches, and $t_i$ denotes the time required for $i$-th quantum search.

To show that the first statement holds, let us recall that the Grover’s algorithm searches the list of $N$ elements to find an element with the desired property. Exhaustive search can find this element by $N$ calls to a procedure which checks whether a given element has this property. While the (worst-case) running time of exhaustive search is $r \cdot N$, where $r$ is the running time of the checking procedure, Grover’s algorithm enables us to find the desired element in $c \cdot \sqrt{N}$ calls to this procedure, where $c$ is a constant determined by the required probability of success. So, the running time of Grover’s algorithm is $r \cdot c \cdot \sqrt{N}$.

In the $i$-th Grover’s search, $t_i = r_i \cdot c \cdot \sqrt{N_i}$, where $N_i$ is the number of elements in the corresponding list and $r_i$ is the running time of the corresponding checking procedure. So, we can conclude that

$$N_i = \frac{t_i^2}{r_i^2 \cdot c^2}.$$ 

Hence, by using (non-quantum) exhaustive search algorithm, we can perform the same search in time

$$t'_i = r_i \cdot N_i = \frac{t_i^2}{r_i \cdot c^2}.$$ 

Since $r_i \geq 1$, we conclude that $t'_i \leq c' \cdot t_i^2$, where $c' = \max(1, c^{-2})$.

Since $t_0$ is a non-negative integer, we have $t_0 \leq t_0^2$; since $c' \geq 1$, we have $t_0 \leq c' \cdot t_0^2$. Thus, by replacing each Grover’s search by the non-quantum search, we get the time $T = t_0 + t'_1 + \ldots + t'_s$. Here, $t'_i \leq c' \cdot t_i^2$ for all $i$, hence $T \leq c' \cdot (t_0^2 + t_1^2 + \ldots + t_s^2)$. Since

$$t_0^2 + \ldots + t_s^2 \leq (t_0 + \ldots + t_s)^2 = t_0^2 + \ldots + t_s^2 + 2 \cdot t_0 \cdot t_1 + \ldots,$$

we conclude that $T \leq c' \cdot T_Q^2$.

**Second statement.** Since $T \leq c' \cdot T_Q^2$, we have $T_Q \geq \frac{1}{c' \cdot \sqrt{T}}$, i.e., $T_Q = \Omega(\sqrt{T})$.

**Remark.** Our observation is valid only if we restrict the use of quantum computation to Grover’s algorithm. There are quantum techniques which lead to a faster speed-up. For example, the well-known Shor’s algorithm for factoring large integers requires polynomial time [23, 24, 15], while all known non-quantum factorization algorithms require, in the worst case, exponential time. If we can use such techniques, we might get more than quadratic speed-up.

**Acknowledgments.** This work was supported in part by NASA under cooperative agreement NCC5-209, NSF grant EAR-0225670, NIH grant 3T34GM008048-20S1, and Army Research Lab grant DATM-05-02-C-0046.
References


