Quantum Versions of k-CSP Algorithms: a First Step Towards Quantum Algorithms for Interval-Related Constraint Satisfaction Problems

Evgeny Dantsin
Alexander Wolpert
Department of Computer Science
Roosevelt University
Chicago, IL 60605, USA
{edantsin,awolpert}@roosevelt.edu

Vladik Kreinovich
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
vladik@utep.edu

ABSTRACT

In data processing, we input the results $\widetilde{x}_i$ of measuring easy-to-measure quantities x_i and use these results to find estimates $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ for difficult-to-measure quantities y which are related to x_i by a known relation $y = f(x_1, \ldots, x_n)$. Due to measurement inaccuracy, the measured values $\widetilde{x}_i$ are, in general, different from the (unknown) actual values x_i of the measured quantities, hence the result $\widetilde{y}$ of data processing is different from the actual value of the quantity y.

In many practical situations, we only know the bounds Δ_i on the measurement errors $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$. In such situations, we only know that the actual value x_i belongs to the interval $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$, and we want to know the range of possible values of y. The corresponding problems of *interval computations* are NP-hard, so solving these problems may take an unrealistically long time. One way to speed up computations is to use quantum computing, and quantum versions of interval computations algorithms have indeed been developed.

In many practical situations, we also know some *constraints* on the possible values of the directly measured quantities $x_1, \ldots, x_n$. In such situations, we must combine interval techniques with constraint satisfaction techniques. It is therefore desirable to extend quantum interval algorithms to such combinations. As a first step towards this combination, in this paper, we consider quantum algorithms for discrete constraint satisfaction problems.

Categories and Subject Descriptors

F.2.1 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Numerical Algorithms and Problems*; G.1.0 [**Mathematics of Computing**]: Numerical Analysis—*Error analysis*; G.4 [**Mathematics of Computing**]: Mathematical Software—*Algorithm design and analysis*

1. INTRODUCTION

1.1 Importance of interval computations

In many practical problems, we are interested in the value of a physical quantity y that is difficult or even impossible to measure directly. Since it is difficult to measure y *directly*, we then measure y *indirectly*, i.e., we measure the values of easier-to-measure quantities $x_1, \ldots, x_n$ which are related to y in a known way $y = f(x_1, \ldots, x_n)$, and then we use the results $\widetilde{x}_i$ of measuring x_i to compute the estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ for the desired quantity y.

Measurements are never 100% accurate; as a result, the measured values $\widetilde{x}_i$ is, in general, different from the actual value x_i of the measured property: $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i \neq 0$. As a result, the estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ differs from the actual value $y = f(x_1, \ldots, x_n)$ of the desired quantity – even when we know the exact algorithm for the dependence $y = f(x_1, \ldots, x_n)$ between x_i and y.

Traditionally in science and engineering, it is assumed that we know the probability of different values of measurement errors Δx_i. These probabilities are usually determined when we *calibrate* the measuring instrument used to measure x_i, i.e., when we compare the results of measuring with this instrument and the results of measuring with a much more accurate *standard* measuring instrument. However, in many real life situations, we do not know these probabilities:

- in state-of-the-art measurements, the instrument we use is the best available; in such situations, there is no better measuring instrument and so, calibration is not possible;
- in manufacturing, calibration is, in principle, possible, but its cost is often much higher than the cost of the sensor itself.

In such cases, instead of the probabilities, we only know the bounds Δ_i on the absolute value of the measurement error provided by the manufacturer of the measuring instrument. In this case, after we get the measurement result $\widetilde{x}_i$, the only information that we have about the (unknown) actual value x_i of the i-th measured quantity is that x_i belongs to the interval $\mathbf{x}_i \stackrel{\text{def}}{=} [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. In this case, we must determine the range

$$\mathbf{y} \stackrel{\text{def}}{=} \{f(x_1, \ldots, x_n) \,|\, x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}$$

of possible values of $y = f(x_1, \ldots, x_n)$. Computing this range is the main problem of *interval computations*; see, e.g., [14].

1.2 The need for quantum algorithms in interval computations and in CSPs

In general, interval computation problems are difficult to solve (NP-hard), even for quadratic functions $f(x_1, \ldots, x_n)$; see, e.g., [15]. This means, in effect, that any algorithm for computing the range will need, in the worst case, computation time which grows exponentially with the number n of inputs. For large n, the resulting computation time becomes unrealistically long.

One way to speed up computations is to use quantum algorithms; see, e.g., [16]. The main attraction of quantum computing is that it can speed up computations. In particular, Grover's quantum algorithm [9, 10, 11, 18] 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.

It is known that in many interval computation problems, constraint satisfaction techniques help [14]. It is also known that in some practical problems, there are known relation between the directly measured quantities $x_1, \ldots, x_n$, so instead of finding the range of the function $f(x_1, \ldots, x_n)$ on the box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$, we must find the range of this function under the corresponding constraints.

It is therefore desirable to extend quantum algorithms described in [16] to the case when we also have constraints.

1.3 Scope of this work

In this paper, we start with the simplest type of constraint satisfaction problems (CSP): discrete CSPs, and we show how quantum algorithms can be used to solve these problems.

We hope that these algorithms will eventually lead to efficient quantum algorithms for solving interval-related continuous CSP problems as well.

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, 17, 19].

2. K-CSP PROBLEMS

In this paper, we will consider discrete *constraint satisfaction problem* (CSP), where each of n variables $x_1, \ldots, x_n$ can take $d \geq 2$ possible values, and the problem is to find the values x_i which satisfy given constraints. A simple exhaustive search can solve this problem in time $\sim d^n$, where $\sim$ means equality modulo a term which is polynomial in the length of the input formula.

In this paper, we will mainly consider k-CSP, in which every constraint contains $\leq k$ variables.

One of the fastest (in terms of proven worst-case complexity) algorithms for k-CSP was proposed by Schöning [24]. Schöning's algorithm is a *multi-start random walk* algorithm that repeats the polynomial-time random walk procedure $\mathcal{S}$ exponentially many times. This procedure $\mathcal{S}$ takes as input a k-CSP problem and does the following:

- Choose an initial assignment a $(x_1 = a_1, \ldots, x_n = a_m)$ uniformly at random.
- Repeat $3n$ times:
    - If all the constraints are satisfied by the assignment a, then return a and halt.
    - Otherwise, pick any constraint which is not satisfied by a; choose one of the $\leq k$ variables x_i involved in this constraints uniformly at random; modify a by changing the chosen variable x_i from its original value to one of the other $d - 1$ values (chosen uniformly at random).

As shown in [24], if the formula F is satisfiable, then each random walk of length $3n$ finds a satisfying assignment with the probability $\geq (d \cdot (1 - 1/k) + \varepsilon)^{-n}$, where ε can be arbitrarily small. Therefore, for any constant probability of success, after $O((d \cdot (1 - 1/k) + \varepsilon)^n)$ runs of the random walk procedure $\mathcal{S}$, we get a satisfying assignment with the required probability. Since $\mathcal{S}$ is a polynomial time procedure, the overall running time of this algorithm is also $T \sim (d \cdot (1 - 1/k) + \varepsilon)^n$. Schöning's algorithm was, in effect, derandomized in [6].

3. SATISFIABILITY: AN IMPORTANT PARTICULAR CASE OF CSP

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. Here, clauses C_j are constraints.

A simple exhaustive search can solve this problem in time $\sim 2^n$.

k-CSP leads to k-SAT, a restricted version of SAT where each clause has at most k literals. For k-SAT, Schöning's algorithm repeats the polynomial-time random walk procedure $\mathcal{S}$ exponentially many times. This procedure $\mathcal{S}$ 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.

The overall running time of this algorithm is $T \sim (2 - 2/k)^n$. This upper bound is close to the best known upper bound for k-SAT (see below).

4. AMBAINIS' OBSERVATION FOR SATISFIABILITY

As we have mentioned, a simple exhaustive search can solve the satisfiability problem in time $\sim N = 2^n$. Grover's algorithm can find the satisfying assignment a in time $O(\sqrt{N})$ with arbitrary high probability of success. (More exactly, this reduction comes from the modification of the original Grover's algorithm called *amplitude amplification* [3, 5].) Thus, a reasonably 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 Schöning's algorithm for 3-SAT [24], there are $N \sim (2 - 2/k)^n$ results of different runs of $\mathcal{S}$, and we look for a result in which the input formula F is satisfied. Thus, Schöning's algorithm can be similarly sped up from time $T \sim (2 - 2/k)^n$ to $\sqrt{T} \sim (2 - 2/k)^{n/2}$; this observation was first made by Ambainis [3].

For 3-SAT, Schöning's algorithm was improved by Rolf [22] 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 $\sqrt{T} \sim 1.154^n$.

5. THE FASTEST ALGORITHM FOR k-SAT

We have mentioned that 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 from SAT to k-CSP [24]. The running time of the corresponding algorithm is $T \sim (d \cdot (1 - 1/k) + \varepsilon)^n$, where ε can be arbitrarily small. Similar to Schöning's algorithm for k-SAT, this extension to k-CSP can be quantized with the running time $T_Q \sim \sqrt{T} \sim (d \cdot (1 - 1/k) + \varepsilon)^{n/2}$. A different quantum algorithm for 2-CSP is described in [4].

The best known upper bound for k-SAT is given by the algorithm proposed by Paturi, Pudlák, Saks, and Zane [20, 21]; 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 [21], the PPSZ algorithm runs in time $T \sim 2^{n \cdot (1 - \mu_k/k)}$, where $\mu_k \to \pi^2/6$ as k increases. The PPSZ algorithm was derandomized in [23] 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 $\sqrt{T} \sim 1.151^n$.

6. 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 [25]. This approach suggests exponentially many runs of the following polynomial-time procedure $\mathcal{S}$:

- 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.
- 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 could guess this clause, we could then 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 truth values which come from the assumption that this clause is false.
- Finally, we recursively apply $\mathcal{S}$ to the result of simplification.

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

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

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

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

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

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

7. ANALYZING POSSIBILITY OF FURTHER 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 $\mathcal{A}_Q$ that solves a problem in time T_Q, then we can "dequantize" it into a non-quantum algorithm $\mathcal{A}$ 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 $T_Q = \Omega(\sqrt{T})$.

7.1 Demonstration of Statement 1

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 $\mathcal{A}_Q$, 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 $T \leq c' \cdot T_Q^2$.

7.2 Demonstration of Statement 2

Since $T \leq c' \cdot T_Q^2$, we have $T_Q \geq (1/\sqrt{c'}) \cdot \sqrt{T}$, 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 [26, 27, 18], 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.

8. CONCLUSION

In many real-life applications, we must solve a constraint satisfaction problem (CSP), i.e., we must find the values $x_1, \ldots, x_n$ of the quantities x_i which satisfy given constraints. In general, such problems are difficult to solve (NP-hard), which means, in effect, that any algorithm for solving the corresponding problem will need, in the worst case, computation time which grows exponentially with the number n of inputs. For large n, the resulting computation time becomes unrealistically long.

One way to speed up computations is to use quantum algorithms which can speed up computations. In particular, Grover's quantum algorithm searches an unsorted list of N elements, and in time $O(\sqrt{N})$, finds an element with a given property. It is therefore desirable to use quantum computers to speed up algorithms for solving CSPs.

In this paper, we consider the simplest type of constraint satisfaction problems: discrete k-CSPs, where each of n variables $x_1, \ldots, x_n$ can take $d \geq 2$ possible values, and every constraint contains $\leq k$ variables. A simple exhaustive search can solve this problem in time $\sim d^n$. Several algorithms have been proposed which solve k-CSP problems faster, with worst-case time complexity $T \ll d^n$. We show that for these algorithms, Grover's technique can reduce the computation time to $T_Q \sim \sqrt{T}$.

A similar reduction can be achieved for algorithms which solve particular cases of discrete CSP problems, such as particular cases of propositional satisfiability. We also demonstrate that if we only use Grover's technique, then we can achieve at most quadratic speed-up: namely, 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 $T_Q = \Omega(\sqrt{T})$.

9. 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. The authors are thankful to the anonymous referees for their helpful suggestions.

10. REFERENCES

[1] S. Aaronson. NP-complete problems and physical reality. *ACM SIGACT News*, 36(1):30–52, 2005.

[2] L. Accardi and M. Ohya. A stochastic limit approach to the SAT problem. *Open Systems and Infomation Dynamics*, 11:219–233, 2004.

[3] A. Ambainis. Quatum search algorithms. *ACM SIGACT News*, 35(2):22–35, 2004.

[4] O. Angelsmark, V. Dahllöf, and P. Jonsson. Finite domain constraint satisfaction using quantum computation, In: K. Diks and W. Rytter (Eds.), *Proceedings of the 27th International Symposium Mathematical Foundations of Computer Science MFCS'2002*, Warsaw, Poland, August 26–30, 2002, volume 2420 of *Lecture Notes in Computer Science*, pages 93–103, 2002.

[5] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. In: *Quantum Computation and Quantum Information Systems*, American Mathematical Society, Contemporary Math Series, 2000, volume 305, pages 53–74.

[6] E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, and U. Schöning. A deterministic $(2-2/(k+1))^n$ algorithm for k-SAT based on local search. *Theoretical Computer Science*, 289:69-83, 2002.

[7] E. Dantsin and A. Wolpert. Derandomization of Schuler's algorithm for SAT. In: *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing SAT'2004*, pages 69–75, 2004; an extended version will appear in Springer Lecture Notes in Computer Science.

[8] E. Dantsin and A. Wolpert. An improved upper bound for SAT, In: F. Bacchus and T. Walsh (eds.), *Proceedings of the 8th International Conference on Theory and Applications on Satisfiability Testing SAT'2005*, volume 3569 of *Lecture Notes in Computer Science*, pages 400–407, 2005.

[9] L. K. Grover. A fast quantum mechanical algorithm for database search. In: *Proceedings of the 28th Symposium on Theory of Computing STOC'96*, ACM Press, New York, pages 212–219, 1996.

[10] L. K. Grover. Quantum mechanics helps in searching for a needle in a haystack. Phys. Rev. Letters, 78(2):325–328, 1997.

[11] L. K. Grover. A framework for fast quantum mechanical algorithms. In *Proceedings of the 30th Symposium on Theory of Computing STOC'98, Dallas, Texas, May 23–26, 1998*, ACM Press, New York, pages 53–62, 1998.

[12] T. Hogg T. Adiabatic quantum computing for random satisfiability problem. Phys. Rev. A, 67:022314, 2003.

[13] K. Iwama and S. Tamaki. Improved upper bounds for 3-SAT. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms SODA'2004*, page 328, 2004.

[14] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer, London, 2001.

[15] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1997.

[16] V. Kreinovich and L. Longpré, Fast quantum algorithms for handling probabilistic and interval uncertainty, *Mathematical Logic Quarterly*, 2004, Vol. 50, No. 4/5, pp. 507–518.

[17] T. Mihara and T. Nishino. On a method of solving SAT efficiently using the quantum Turing machine. In *Proceedings of the Workshop on Physics and Computation, Dallas, Texas, November 17–20, 1994*, pages 177–185, 1994.

[18] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information.* Cambridge University Press, Cambridge, U.K., 2000.

[19] M. Ohya. Quantum algorithm for SAT problem and quantum mutual entropy. In *Proceedings of the von Neumann Centennial Conference: Linear Operators and Foundations of Quantum Mechanics, Budapest, Hungary, 15–20 October, 2003*, 2003.

[20] R. Paturi, P. Pudlák, and F. Zane. Satisfiability coding lemma. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science FOCS'97*, pages 566–574, 1997.

[21] R. Paturi, P. Pudlák, S. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. In *Proceedings of the 39th IEEE Conference on Foundations of Computer Science FOCS'98*, pages 628–637, 1998.

[22] D. Rolf. 3-SAT in $RTIME(O(1.32971^n))$ — improving randomized local search by initializing strings of 3-clauses. *Electronic Colloquium on Computational Complexity*, Report No. 54, July 2003.

[23] D. Rolf. Derandomization of PPSZ for Unique-k-SAT. In: F. Bacchus and T. Walsh (eds.), *Proceedings of the 8th International Conference on Theory and Applications on Satisfiability Testing SAT'2005*, volume 3569 of *Lecture Notes in Computer Science*, pages 216–225, 2005.

[24] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proceedings of the 40th Annual Symposium on Fundamentals of Computer Science FOCS'99*, pages 410–414, 1999.

[25] R. Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. Journal of Algorithms, 54(1):40–44, 2005.

[26] P. W. Shor. Algorithms for quantum computations: discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Fundamentals of Computer Science FOCS'94*, pages 124–134, 1994.

[27] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Computing*, 26(5):1484–1509, 1997.