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

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

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

- **Data processing:**
  - we input the results $\tilde{x}_i$ of measuring easy-to-measure quantities $x_i$, and
  - we use these results to find estimates $\tilde{y} = f(\tilde{x}_1, \ldots, \tilde{x}_n)$ for difficult-to-measure quantities $y$ which are related to $x_i$ by a known relation $y = f(x_1, \ldots, x_n)$.

- **Interval uncertainty:** often, we only know the bounds $\Delta_i$ on the measurement errors $\Delta x_i \overset{\text{def}}{=} \tilde{x}_i - x_i$, i.e., we only know that the actual value $x_i$ belongs to the interval $[\tilde{x}_i - \Delta_i, \tilde{x}_i + \Delta_i]$.

- **Problem:** we want to know the range of possible values of $y$.

- **Why quantum computing:** this problem is NP-hard; one way to speed up computations is to use quantum computing.

- Quantum interval techniques have indeed been proposed.

- **Constraints:** often, we also know some constraints on the possible values of the directly measured quantities $x_1, \ldots, x_n$.

- **Ultimate objective:** extend quantum interval algorithms to such constraints.

- **In this paper:** as a first step, we consider quantum algorithms for discrete constraint satisfaction problems.
2. **General Problem of Data Processing under Uncertainty**

- *Indirect measurements:* way to measure $y$ that are difficult (or even impossible) to measure directly.
- *Idea:* $y = f(x_1, \ldots, x_n)$

\[
\begin{array}{cc}
\tilde{x}_1 & f \\
\tilde{x}_2 & \\
\vdots & \\
\tilde{x}_n & \tilde{y} = f(\tilde{x}_1, \ldots, \tilde{x}_n)
\end{array}
\]

- *Problem:* measurements are never 100% accurate: $\tilde{x}_i \neq x_i$ ($\Delta x_i \neq 0$) hence

\[
\tilde{y} = f(\tilde{x}_1, \ldots, \tilde{x}_n) \neq y = f(x_1, \ldots, y_n).
\]

What are bounds on $\Delta y \overset{\text{def}}{=} \tilde{y} - y$?
3. Probabilistic and Interval Uncertainty

- **Traditional approach:** we know probability distribution for $\Delta x_i$ (usually Gaussian).

- **Where it comes from:** calibration using standard MI.

- **Problem:** sometimes we do not know the distribution because no “standard” (more accurate) MI is available. Cases:
  - fundamental science
  - manufacturing

- **Solution:** we know upper bounds $\Delta_i$ on $|\Delta x_i|$ hence
  $$x_i \in [\tilde{x}_i - \Delta_i, \tilde{x}_i + \Delta_i].$$
4. Interval Computations: A Problem

- **Given:**
  - an algorithm $y = f(x_1, \ldots, x_n)$ that transforms $n$ real numbers $x_i$ into a number $y$;
  - $n$ intervals $x_i = [x_i, \bar{x}_i]$.

- **Compute:** the corresponding range of $y$:
  \[
  [y, \bar{y}] = \{f(x_1, \ldots, x_n) \mid x_1 \in [x_1, \bar{x}_1], \ldots, x_n \in [x_n, \bar{x}_n]\}.
  \]

- **Fact:** even for quadratic $f$, the problem of computing the exact range $y$ is NP-hard.

- **Practical challenge:** speed up interval computations.
5. Additional Problem: Constraints

- Traditional interval computations:
  - we know the intervals $x_i$ of possible values of different parameters $x_i$, and
  - we assume that an arbitrary combination of these values is possible.

- In geometric terms: the set of possible combinations $x = (x_1, \ldots, x_n)$ is a box $x = x_1 \times \ldots \times x_n$.

- In practice: we also know additional restrictions on the possible combinations of $x_i$.

- Example: in geosciences, in addition to intervals for velocities $v_i$ at different points, we know that $|v_i - v_j| \leq \Delta$ for neighboring points:

- Example: in nuclear engineering, experts often state that combinations of extreme values are impossible, we have an ellipsoid, not a box.
6. The Need for Quantum Algorithms in Interval Computations and in CSPs

- **Problem:** interval computation problems are difficult to solve (NP-hard).

- **In plain words:** computation time grows exponentially with the number \( n \) of inputs.

- **Result:** For large \( n \), the resulting computation time is unrealistically long.

- **Quantum algorithms:** a way to speed up computations.

- **Example:** Grover’s algorithm searches an unsorted list of \( N \) elements in time \( O(\sqrt{N}) \).

- **What is known:** quantum algorithms for (pure) interval computation.

- **Ultimate objective:** efficient quantum algorithms for solving interval-related continuous CSP problems.

- **In this paper:** we show how quantum computing can speed up the simplest constraint satisfaction problems (CSP): discrete CSPs.
7. **$k$-CSP problems**

- **Discrete CSP:**
  - each of $n$ variables $x_1, \ldots, x_n$ can take $d \geq 2$ possible values, and
  - the goal is to find the values $x_i$ which satisfy given constraints.

- **Exhaustive search:** solves this problem in time $\sim d^n$ ($\sim$ means equality modulo a term which is polynomial in the length of the input formula).

- **Important case:** $k$-CSP problems, in which every constraint contains $\leq k$ variables.

- **SAT:** another important case of CSP is 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 $l_1 \lor \ldots \lor l_k$ 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.
8. **Known Algorithm for $k$-CSP**

- **Known:** one of the fastest (in terms of proven worst-case complexity) Schöning’s multi-start random walk algorithm.

- **Description:** this algorithm repeats the following polynomial-time random walk procedure $S$ exponentially many times:
  
  - 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 $x_i$ from 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).

- For any constant probability of success, after $O((d \cdot (1 - 1/k) + \varepsilon)^n)$ runs of the random walk procedure $S$, we get a satisfying assignment with the required probability.

- **Comment:** there exists a derandomized version of this algorithm.
9. Schöning’s Algorithm for Satisfiability

- For $k$-SAT, Schöning’s algorithm repeats the following polynomial-time random walk procedure $S$ exponentially many times:
  
  - 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 $T \sim (2 - 2/k)^n$.

- Quantum version:
  
  - in Schöning’s algorithm, we search among $N \sim (2 - 2/k)^n$ results of running $S$;
  
  - Grover’s quantum search can thus speed it up from time $T \sim (2 - 2/k)^n$ to $\sqrt{T} \sim (2 - 2/k)^{n/2}$.

- Comment:
  
  - for 3-SAT, Rolf improved this algorithm to $T \sim 1.330^n$;
  
  - this improvement also consists of exponentially many runs of a polynomial-time algorithm;
  
  - thus, Rolf’s non-quantum time $T \sim 1.330^n$ leads to the quantum time $\sqrt{T} \sim 1.154^n$. 

10. The Fastest Known Algorithm for $k$-SAT: PPSZ (Paturi, Pudlák, Saks, and Zane)

- This algorithm consists of exponentially many runs of the following polynomial-time procedure:
  - 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.

- PPSZ runs in time $T \sim 2^n \cdot (1 - \mu_k/k)$, where $\mu_k \to \pi^2/6$ as $k$ increases.

- Grover’s technique leads to a quantum version which requires time $T_Q \sim \sqrt{T}$.

- Comment: for 3-SAT, Iwama and Tamaki proposed a $T \sim 1.324^n$ modification of PPSZ.

- Grover’s algorithm can reduce this to $\sqrt{T} \sim 1.151^n$. 

11. The Fastest Algorithm for SAT with No Restriction on Clause Length (Danstine and Wolpert)

- This approach consists of exponentially many runs of the following polynomial-time procedure $S$:
  - For each clause $C_j$ longer than $k$, we keep the first $k$ literals (and delete the other literals).
  - We use one random walk of Schöning’s algorithm to test satisfiability of the resulting $k$-SAT formula $F'$.
  - If the resulting assignment $a$ satisfies $F$, we are done.
  - Otherwise:
    * we choose a clause in $F'$ at random and assume that this clause is false;
    * we replace the variables in $F$ with the truth values which come from this assumption.
    * we (recursively) apply $S$ to the result of this replacement.

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

- Grover’s technique leads to
  \[
  T_Q \sim \sqrt{T} \sim 2^{-\left(\frac{n}{2}\right)} \left(1 - \frac{1}{\ln \left( \frac{m}{n} \right) + O(\ln \ln (m))} \right).
  \]
12. Analyzing Possibility of Further Speed-Up

• *What we did so far:* we used Grover’s technique to speed up the non-quantum computation time $T$ to the quantum computation time $T_Q \sim \sqrt{T}$.

• *Additional result:* if we only use Grover’s technique, then we cannot get a further time reduction.

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

• **Statement 2.**
  
  – *Assumption:* we have a non-quantum algorithm that solves a problem in time $T$.
  
  – *Conclusion:* any Grover-based quantum algorithm for solving this problem requires time at least $T_Q = \Omega(\sqrt{T})$.

• Proof: in the Proceedings.
13. Conclusion

- Constraint satisfaction problems (CFP) are important in many real-life applications.

- In general, such problems are difficult to solve (NP-hard) – any algorithm will need 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.

- In particular, Grover’s quantum algorithm searches an unsorted list of $N$ elements in time $O(\sqrt{N})$.

- 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 solves this problem in time $\sim d^n$.

- Several algorithms solve $k$-CSP problems in time $T \ll d^n$.

- What we show: for known algorithms, Grover’s technique can reduce the computation time to $T_Q \sim \sqrt{T}$. 
14. Acknowledgments

This work was supported in part:

- by NASA under cooperative agreement NCC5-209,
- by NSF grant EAR-0225670,
- by NIH grant 3T34GM008048-20S1,
- by Army Research Lab grant DATM-05-02-C-0046,
- by Star Award from the University of Texas System,
- by Texas Department of Transportation grant No. 0-5453.