How Quantum Computing Can Help With (Continuous) Optimization

Christian Ayub, Martine Ceberio, and Vladik Kreinovich

Department of Computer Science
University of Texas at El Paso, El Paso, Texas 79968, USA,
cayub@miners.utep.edu, mceberio@utep.edu, vladik@utep.edu

1. Known Advantages of Quantum Computing

- It is known that quantum computing enables us to drastically speed up many computations.
- One example of such a problem is the problem of looking for a given element in an unsorted n-element array.
- With non-quantum computations:
  - to be sure that we have found this element,
  - we need to spend at least n computational steps.
- Indeed, if we use fewer than n steps:
  - this would mean that we only look at less than n elements of the array, and thus,
  - we may miss the element that we are looking for.

2. Advantages of Quantum Computing (cont-d)

- Grover's quantum-computing algorithm allows us to reduce the time to $c \cdot \sqrt{n}$.
- So, we reduce the non-quantum computation time T to

$$T_q \sim \sqrt{T}.$$

3. Need to Consider Optimization Problems

- In many applications, we also need to solve continuous optimization problems.
- We want to find an object or a strategy for which the given objective function attains its maximum.
- An object is usually characterized by its parameters

$$x_1, \ldots, x_n.$$

- For each x_i, we usually know the bounds: $\underline{x}_i \le x_i \le \overline{x}_i$.
- Let $f(x_1, \ldots, x_n)$ denote the value of the objective function corresponding to the parameters $x_1, \ldots, x_n$.
- In most practical situations, the objective function is several (at least two) times continuously differentiable.

4. General Optimization Problem

- Ideal case: find $x_1^{\rm opt}, \ldots, x_n^{\rm opt}$ for which $f(x_1, \ldots, x_n)$ attains its maximum on the box

$$B \stackrel{\rm def}{=} [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n].$$

- In practice, we can only attain values approximately.
- So, in practice, we are looking for the values $x_1^d, \ldots, x_n^d$ which are maximal with given accuracy $\varepsilon > 0$:

$$f(x_1^d, \ldots, x_n^d) \geq \left(\max_{(x_1, \ldots, x_n) \in B} f(x_1, \ldots, x_n) \right) - \varepsilon.$$

- We show that for this problem, quantum computing reduced computation time T to $T_q \sim \sqrt{T} \cdot \ln(T)$.

5. We Consider Only Guaranteed Global Optimization Algorithms

- Of course, there are many semi-heuristic ways to solve the optimization problem.
- For example, we can start at some point $x = (x_1, \ldots, x_n)$ and use gradient techniques to reach a *local* maximum.
- However, these methods only lead to a local maximum.
- If we want to make sure that we reached the actual (*global*) maximum:
  - we cannot skip some subdomains of the box B,
  - we have to analyze all of them.
- How can we do it?

6. Non-Quantum Lower Bound

- Let us select some size $\delta > 0$ (to be determined later).
- Let us divide each interval $[\underline{x}_i, \overline{x}_i]$ into $N_i \stackrel{\text{def}}{=} \dfrac{\overline{x}_i - \underline{x}_i}{\delta}$ subintervals of width δ.
- This divides the whole box B into into

$$N = N_1 \cdot \ldots \cdot N_n = \prod_{i=1}^{n} \frac{\overline{x}_i - \underline{x}_i}{\delta} = \frac{V}{\delta^n} \text{ subboxes.}$$

- Here, V is the volume of the original box B:

$$V \stackrel{\text{def}}{=} (\overline{x}_1 - \underline{x}_1) \cdot \ldots \cdot (\overline{x}_n - \underline{x}_n).$$

- We can have functions which are 0 everywhere except for one subbox at which this function grows to $1.1 \cdot \varepsilon$.
- On this subbox, the function is approximately quadratic.
- We have a bound S on the second derivative.

Home Page
Title Page
◀◀ ▶▶
◀ ▶
Page 7 of 24
Go Back
Full Screen
Close
Quit

7. Non-Quantum Lower Bound (cont-d)

- This function starts with 0 at a neighboring subbox.
- So, it cannot grow faster than $S \cdot x^2$ on this subbox.
- Thus, to reach a value larger than ε, we need to select δ for which $S \cdot (\delta/2)^2 = 1.1 \cdot \varepsilon$, i.e., the value $\delta \sim \varepsilon^{1/2}$.
- For this value δ, we get $V/\delta^n \sim \varepsilon^{-(n/2)}$ subboxes:
  - if we do not explore some values of the optimized function at each of the subboxes,
  - we may miss the subbox that contains the largest value.
- Thus, we will not be able to localize the point at which the function attains its maximum.
- So, to locate the global maximum, we need at least as many computation steps as there are subboxes.
- So, we need at least time $\sim \varepsilon^{-(n/2)}$.

8. This Lower Bound Is Reachable

- Let us show that there exists an algorithm that always locates the global maximum in time $\sim \varepsilon^{-(n/2)}$.
- Let us divide the box B into subboxes of linear size δ.
- For each subbox b, each of its sides has size $\le \delta$.
- Thus, each component x_i differs from the midpoint's $x^m \stackrel{\text{def}}{=} (x_1^m, \ldots, x_n^m)$ by no more than $\delta/2$:

$$|\Delta x_i| \le \delta/2, \text{ where } \Delta x_i \stackrel{\text{def}}{=} x_i - x_i^m.$$

- Thus, by using known formulas from calculus, we can conclude that for each point $x = (x_1, \ldots, x_n) \in b$:

$$f(x_1, \ldots, x_n) = f(x_1^m + \Delta x_1, \ldots, x_n^m + \Delta x_n) =$$

$$f(x_1^m, \ldots, x_n^m) + \sum_{i=1}^n c_i \cdot \Delta x_i + \sum_{i=1}^n \sum_{j=1}^n c_{ij} \cdot \Delta x_i \cdot \Delta x_j.$$

9. This Lower Bound Is Reachable (cont-d)

- Here $c_i \stackrel{\text{def}}{=} \dfrac{\partial f}{\partial x_i}(x_1^m, \ldots, x_n^m)$, and $c_{ij} \stackrel{\text{def}}{=} \dfrac{\partial^2 f}{\partial x_i \partial x_j}(\xi_1, \ldots, \xi_n)$ for some $\xi_1, \ldots, \xi_n) \in b$.
- We assumed that the function f is twice continuously differentiable.
- So, all its second derivatives are continuous.
- Thus, there exists a general bound S on all the values of all second derivatives: $|c_{ij}| \le S$.
- Because of these bounds, the quadratic terms in the above formula are bounded by $n^2 \cdot S \cdot (\delta/2)^2 = O(\delta^2)$.

10. These Estimations Lead to a Global Piece-Wise Linear Approximate Function

- By considering only linear terms on each subbox, we get an approximate piece-wise linear function $f_{\approx}(x_1, \ldots, x_n)$.
- On each subbox b:

$$f_{\approx}(x_1, \ldots, x_n) = f(x_1^m, \ldots, x_n^m) + \sum_{i=1}^{n} c_i \cdot \Delta x_i.$$

- For each $x = (x_1, \ldots, x_n)$, we have

$$|f(x_1, \ldots, x_n) - f_{\approx}(x_1, \ldots, x_n)| \le n^2 \cdot S \cdot (\delta/2)^2.$$

11. Optimizing the Approximate Function

- Let us find the point at which linear function $f_\approx(x_1, \ldots, x_n)$ attains its maximum.
- On $[x_1^m - \delta/2, x_1^m + \delta/2] \times \ldots \times [x_n^m - \delta/2, x_n^m + \delta/2]$, as one can easily see:
  - the function $f_\approx(x_1, \ldots, x_n)$ is increasing with respect to each x_i when $c_i \geq 0$ and
  - the function $f_\approx(x_1, \ldots, x_n)$ is decreasing with respect to x_i if $c_i \leq 0$.
- Thus:
  - when $c_i \geq 0$, the maximum of the function $f_\approx(x_1, \ldots, x_n)$ on this subbox is attained when $x_i = x_i^m + \delta/2$,
  - when $c_i \leq 0$, the maximum of the function $f_\approx(x_1, \ldots, x_n)$ on this subbox is attained when $x_i = x_i^m - \delta/2$.
- We can combine both cases by saying that the maximum is attained when $x_i = x_i^m + \text{sign}(c_i) \cdot (\delta/2)$.

12. Optimizing $f_{\approx}(x)$ (cont-d)

- Here $\text{sign}(x)$ is the sign of x (i.e., 1 if $x \ge 0$ and -1 otherwise).
- We can:
  - repeat this procedure for each subbox,
  - find the corresponding largest value on each subbox, and then
  - find the largest of these values.
- This largest value is attained at a point $x^M = (x_1^M, \ldots, x_n^M)$.
- So, here is where $f_{\approx}(x_1, \ldots, x_n)$ attains its maximum.

13. Proof of Correctness

- Let us show that the point $x^M = (x_1^M, \ldots, x_n^M)$ is indeed a solution to the given optimization problem.
- Indeed, let $x^{\text{opt}} = (x_1^{\text{opt}}, \ldots, x_n^{\text{opt}})$ be a point where the original function $f(x_1, \ldots, x_n)$ attains its maximum.
- Since $f_\approx(x)$ attains its maximum at x^M, we have

$$f_\approx(x^M) \geq f_\approx(x) \text{ for all } x.$$

- In particular, $f_\approx(x_1^M, \ldots, x_n^M) \geq f_\approx(x_1^{\text{opt}}, \ldots, x_n^{\text{opt}})$.
- The functions $f_\approx(x_1, \ldots, x_n)$ and $f(x_1, \ldots, x_n)$ are η-close, where $\eta \stackrel{\text{def}}{=} n^2 \cdot S \cdot (\delta/2)^2$.
- So, $f(x_1^M, \ldots, x_n^M) \geq f_\approx(x_1^M, \ldots, x_n^M) - \eta$ and

$$f_\approx(x_1^{\text{opt}}, \ldots, x_n^{\text{opt}}) \geq f(x_1^{\text{opt}}, \ldots, x_n^{\text{opt}}) - \eta = M - \eta.$$

14. Proof of Correctness (cont-d)

- From these inequalities, we conclude that

$$f(x_1^M, \ldots, x_n^M) \geq f_{\approx}(x_1^M, \ldots, x_n^M) - \eta \geq$$

$$f_{\approx}(x_1^{\text{opt}}, \ldots, x_n^{\text{opt}}) - \eta \geq (M - \eta) - \eta = M - 2\eta.$$

- So, $f(x_1^M, \ldots, x_n^M) \geq M - 2\eta$.
- Thus, for $\eta = \varepsilon/2$, we indeed get a solution to the original problem.

15. How Much Computation Time Do We Need

- To solve the original problem with a given ε, we need to select the value δ for which

$$2n^2 \cdot S \cdot (\delta/2)^2 = \varepsilon.$$

- So, we need $\delta = c \cdot \varepsilon^{1/2}$, for an appropriate constant c.
- In this algorithm, we divide the whole box B of volume V into V/δ^n subboxes of linear size δ.
- Since $\delta \sim \varepsilon^{1/2}$, the overall number of subboxes is proportional to $\varepsilon^{-n/2}$.
- On each subbox, the number of computational steps does not depend on ε.
- So, the overall computation time is proportional to the number of boxes, i.e.,

$$T = \text{const} \cdot \varepsilon^{-n/2}.$$

16. How Quantum Computing Can Help: Preliminary Step

- We have bounded the function on a subbox.
- Similarly, we can find the bounds $\underline{F}$ and $\overline{F}$ on $f(x_1, \ldots, x_n)$ over the whole box B.
- These bounds also bound the maximum M of the function $f(x_1, \ldots, x_n)$: $M \in [\underline{F}, \overline{F}]$.
- By selecting an appropriate $\delta \sim \varepsilon^{1/2}$, we can get an approximate function $f_{\approx}(x)$ which is $(\varepsilon/4)$-close to $f(x)$.
- Because of this closeness, the maximum $M_{\approx}$ of the approximate function is $(\varepsilon/2)$-close to the maximum M.
- Thus, $M_{\approx} \in [\underline{A}_0, \overline{A}_0]$, where

$$\underline{A}_0 \stackrel{\text{def}}{=} \underline{F} - \varepsilon/2 \text{ and } \overline{A}_0 \stackrel{\text{def}}{=} \overline{F} + \varepsilon/2.$$

17. Auxiliary Quantum Algorithm and Its Use

- For each A, Grover's algorithm finds, in time $\sim \sqrt{N}$:
  - one of N subboxes at which the maximum of $f_{\approx}(x)$ on this subbox is larger than or equal to A
  - or that there is no such subbox.
- Let us assume that we know an interval $[\underline{A}, \overline{A}]$ that contains the maximum $M_{\approx}$ of $f_{\approx}$.
- Let's use the above algorithm for $A = (\underline{A} + \overline{A})/2$.
- If there is a subbox b for which $f_{\approx}(x_0) \geq A$ for some $x_0 \in b$, then $M_{\approx} = \max\limits_{x \in B} f_{\approx}(x) \geq f(x_0) \geq A$, so $M_{\approx} \in [A, \overline{A}]$.
- If no such subbox exists, then $f_{\approx}(x) \leq A$ for all x, so $M_{\approx} \leq A$ and $M_{\approx} \in [\underline{A}, A]$.
- In both cases, we get a half-size interval containing $M_{\approx}$.

Known Advantages of . . .
Need to Consider . . .
We Consider Only . . .
Non-Quantum Lower . . .
This Lower Bound Is . . .
How Quantum . . .
Auxiliary Quantum . . .
Main Algorithm: First . . .
Main Algorithm: . . .
Home Page
Title Page
Page 18 of 24
Go Back
Full Screen
Close
Quit

18. Main Algorithm: First Part

- We start with the interval $[\underline{A}, \overline{A}] = [\underline{A}_0, \overline{A}_0]$ that contains the actual value $M_\approx$.
- At each iteration, we apply the above idea with

$$A = (\underline{A} + \overline{A})/2.$$

- As a result, we come up with a half-size interval containing $M_\approx$.
- In k steps, we decrease the width of the interval 2^k times, to $2^{-k} \cdot (\overline{A} - \underline{A})$.
- In particular, in $k \approx \ln(\varepsilon)$, we can get an interval $[\underline{a}, \overline{a}]$ containing $M_\approx$ whose width is $\leq \varepsilon/4$: $\underline{a} \leq M_\approx \leq \overline{a}$.

19. Main Algorithm: Second Part

- Since $\underline{a}$ is $\leq$ the maximum $M_\approx$ of $f_\approx(x_1, \ldots, x_n)$, one of the values of this approximate function is indeed $\geq \underline{a}$.
- The above auxiliary quantum algorithm will then find, in time $\sim \sqrt{N}$, a point $x^q = (x_1^q, \ldots, x_n^q)$ for which

$$f_\approx(x_1^q, \ldots, x_n^q) \geq \underline{a}.$$

20. Proof of Correctness

- Let us prove that the resulting point $x^q = (x_1^q, \ldots, x_n^q)$ indeed solves the original optimization problem.
- Indeed, by the very construction of this point, the value $f_\approx(x^q)$ is greater than or equal to $\underline{a}$.
- Since $f_\approx(x^q)$ cannot exceed the maximum value $M_\approx$ of $f_\approx(x)$, and $M_\approx \le \overline{a}$, we conclude that $f_\approx(x^q) \le \overline{a}$.
- Thus, both $f_\approx(x^q)$ and $M_\approx$ belong to the same interval $[\underline{a}, \overline{a}]$ of width $\le \varepsilon/4$.
- So, the value $f_\approx(x^q)$ is $(\varepsilon/4)$-close to the maximum $M_\approx$.

21. Proof of Correctness (cont-d)

- In particular, this implies that

$$f_{\approx}(x_1^q, \ldots, x_n^q) \ge M_{\approx} - \varepsilon/4.$$

- Since $f_{\approx}(x_1, \ldots, x_n)$ and $f(x_1, \ldots, x_n)$ are $(\varepsilon/4)$-close, their maximum values $M_{\approx}$ and M are also $(\varepsilon/4)$-close.
- In particular, this implies that $M_{\approx} \ge M - \varepsilon/4$, hence

$$f(x_1^q, \ldots, x_n^q) \ge M - \varepsilon/2.$$

- Since the functions are $(\varepsilon/4)$-close, we conclude that $f(x_1^q, \ldots, x_n^q) \ge f_{\approx}(x_1^q, \ldots, x_n^q) - \varepsilon/4$ and thus, that

$$f(x_1^q, \ldots, x_n^q) \ge f_{\approx}(x_1^q, \ldots, x_n^q) - \varepsilon/4 \ge$$

$$(M - \varepsilon/2) - \varepsilon/4 > M - \varepsilon.$$

- Thus, we indeed get the desired solution to the optimization problem.

22. What is the Computational Complexity of This Quantum Algorithm

- We need $\sim \ln(\varepsilon)$ iterations each of which requires time

$$\sim \sqrt{N} \sim \sqrt{\varepsilon^{-(n/2)}} = \varepsilon^{-(n/4)}.$$

- Thus, the overall computation time T_q of this quantum algorithm is equal to $T_q \sim \varepsilon^{-(n/4)} \cdot \ln(\varepsilon)$.
- We know that the computation time T of the non-quantum algorithm is $T \sim \varepsilon^{-(n/2)}$; thus, $\varepsilon^{-(n/4)} \sim \sqrt{T}$.
- Here, $\varepsilon \sim T^{-(2/n)}$, and thus, $\ln(\varepsilon) \sim \ln(T)$.
- Thus, we conclude that

$$T_q \sim \sqrt{T} \cdot \ln(T).$$

- The main result is thus proven.

23. Acknowledgments

This work was supported in part by the US National Science Foundation grant HRD-1242122 (Cyber-ShARE).