Optimization problems on quantum computers

Lecture 2

Quantum optimization algorithms using oracles

Yassine Hamoudi

Materials: https://yassine-hamoudi.github.io/cemracs2025/

Lectures

- *Lecture 1:* Quantum optimization algorithms inspired by physics [Slides]
- *Lecture 2:* Quantum optimization algorithms using oracles [Slides]

Suggested references

- *Lectures:* R. de Wolf [Lecture notes] (Chapter 19.4) [Video], A. Childs [Lecture notes] (Part VI), A. Montanaro [Video 1] [Video 2]
- *Surveys and Textbooks:*
  - "Challenges and Opportunities in Quantum Optimization". A. Abbas et al. [Link]
  - "Quantum Algorithms: A Survey of Applications and End-to-end Complexities". A. Dalzell et al. [Link]
  - "Quantum Algorithms for Optimizers". G. Nannicini. [Link]
  - "Adiabatic Quantum Computing". T. Albash, D. Lidar. [Link]
  - "Variational Quantum Algorithms". M. Cerezo et al. [Link]

https://yassine-hamoudi.github.io/cemracs2025/

Yesterday's lecture:

- very general algorithms applying to broad class of problems
- running time often unknown but expected to be significantly smaller than that of classical methods
- sometimes adapted to near-term and hybrid quantum computers

Today's lecture:

- algorithms that are more computer science–oriented and problem-specific
- precise running time guarantees and provable quantum advantages, but often moderate speedups
- require large-scale fault-tolerant quantum computers

Quantum optimization algorithms

Lecture 1 (Physics-inspired)	Lecture 2 (Oracle-based)
Exact algorithms - Quantum Phase Estimation (QPE) - Quantum Adiabatic Algorithm (QAA)	**Grover-type algorithms** - Quantum Minimum Finding - Minimum Spanning Tree
Variational quantum algorithms - Variational Quantum Eigensolver (VQE) - Quantum Approximate Optimization Algorithm (QAOA)	**Gradient computation** **Monte-Carlo algorithms** - Linear programming - Escaping Saddle Points

Quantum oracles

The algorithms presented in this lecture require a specific input-access model known as a quantum oracle.

Unknown function

i	$F(i)$
1	34
2	12
3	2
4	200

An oracle is a unitary operator U_F that provides a way to evaluate a function on any superposition of values

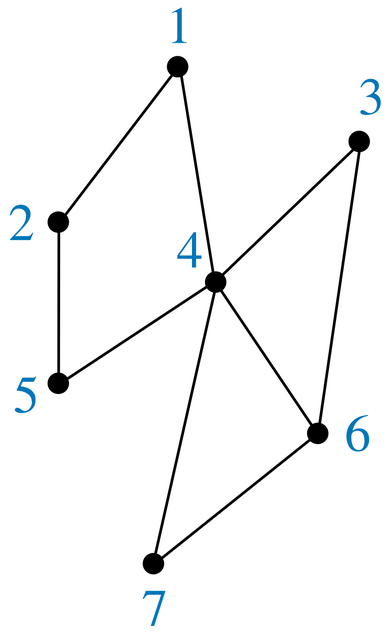
$$U_F : \sum_i \alpha_i |i\rangle |0\rangle \mapsto \sum_i \alpha_i |i\rangle |F(i)\rangle$$

On classical computers, the oracle can only return a single value at a time $i \mapsto F(i)$

We aim at minimizing the number of calls (= queries) to the oracle

Example

Function that enumerates the edges of an unknown graph



i	$F(i)$
1	(1,2)
2	(2,5)
3	(1,4)
4	(3,4)
5	(3,6)
6	(4,6)
7	(4,5)
8	(4,7)
9	(6,7)

How many calls to the oracle are needed to solve a given problem on an unknown graph?

(Ex: Is the graph connected? What is the size of the maximum cut? Etc.)

Example: Bernstein–Vazirani algorithm

Bernstein–Vazirani problem: Oracle to $f : \{0,1\}^d \to \{0,1\}$

$$x \mapsto a_1 x_1 + \ldots + a_d x_d \bmod 2$$

Compute $(a_1, \ldots, a_d)$

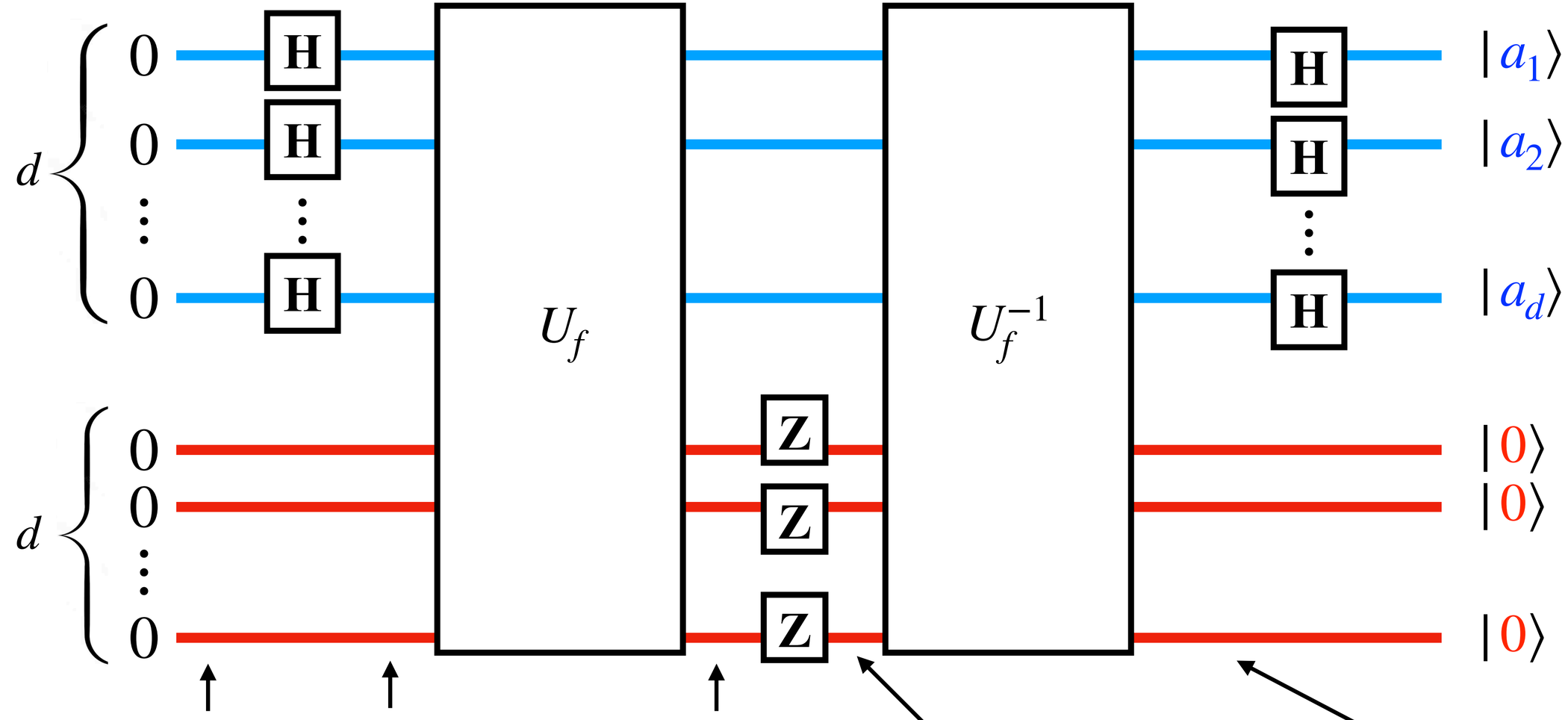
Using a classical oracle

d queries

Using a quantum oracle

2 queries

Example: Bernstein–Vazirani algorithm



$$|0\ldots0, 0\ldots0\rangle$$

$$\frac{1}{2^{d/2}}\sum_{x=0}^{2^d-1}|x, 0\ldots0\rangle$$

$$\frac{1}{2^{d/2}}\sum_{x=0}^{2^d-1}|x, f(x)\rangle$$

$$\frac{1}{2^{d/2}}\sum_{x=0}^{2^d-1}(-1)^{f(x)}|x, f(x)\rangle$$

$$\frac{1}{2^{d/2}}\sum_{x=0}^{2^d-1}(-1)^{f(x)}|x, 0,\ldots,0\rangle$$

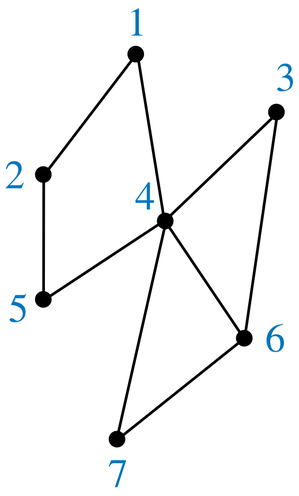
Grover-type algorithms

Grover's algorithm

What it does:

Search in the domain of an oracle if an element satisfies a given predicate

Example



i	$F(i)$
1	(1,2)
2	(2,5)
3	(1,4)
4	(3,4)
5	(3,6)
6	(4,6)
7	(4,5)
8	(4,7)
9	(6,7)

Is there an edge containing vertex 3?

Classical algorithms may have to query all edges

Grover requires quadratically less queries to the oracle

Grover's algorithm

What it does:

Search in the domain of an oracle if an element satisfies a given predicate

… and if so, return a uniform superposition over all the solution elements:

$$\frac{1}{\sqrt{|I|}} \sum_{i \in I} |i\rangle \quad \text{where} \quad I = \{i \,:\, \mathrm{Predicate}(i, F(i)) \,=\, \mathrm{True}\}$$

Grover's algorithm requires $\sim \sqrt{N/K}$ quantum queries, where N is the domain size of F and $K = |I|$ the number of solutions.

Quantum Minimum Finding

Finding the smallest element

x	$f(x)$
1	6
2	2
3	9
4	8
5	3
6	1
7	8
$N = 8$	4

If the function is completely arbitrary, the best possible classical algorithm is to evaluate all entries in time N

How Grover's algorithm can help here?

Set $(x, y) = (1, f(1))$

Repeat:

Prepare with Grover the superposition over all x' satisfying $f(x') < y$.

Sample x' s.t. $f(x') < y$, uniformly at random, by measuring the state.

Update $(x, y) = (x', f(x'))$.

Finding the smallest element

x	$f(x)$
1	6
2	2
3	9
4	8
5	3
6	1
7	8
8	4

If the function is completely arbitrary, the best possible classical algorithm is to evaluate all entries in time N

How Grover's algorithm can help here?

Set $(x, y) = (1, f(1))$

Repeat:

Prepare with Grover the superposition over all x' satisfying $f(x') < y$.

Sample x' s.t. $f(x') < y$, uniformly at random, by measuring the state.

Update $(x, y) = (x', f(x'))$.

Finding the smallest element

x	$f(x)$
1	6
2	2
3	9
4	8
5	3
6	1
7	8
8	4

If the function is completely arbitrary, the best possible classical algorithm is to evaluate all entries in time N

How Grover's algorithm can help here?

Set $(x, y) = (1, f(1))$

Repeat:

Prepare with Grover the superposition over all x' satisfying $f(x') < y$.

Sample x' s.t. $f(x') < y$, uniformly at random, by measuring the state.

Update $(x, y) = (x', f(x'))$.

Converges to the minimum in $\log N$ steps

The overall query complexity is $\sim \sqrt{N}$.

Example: MAX-SAT

Find an assignment that maximizes the number of satisfied clauses in a CNF formula

$$f(x) = \neg x_1 \wedge (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge x_3 \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$$

x	$-f(x)$
000	- 3
001	- 4
010	- 4
011	**- 5**
100	- 2
101	- 3
110	- 3
111	- 3

Compiles the formula into a quantum circuit that simulates the oracle $U_f : |x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle$

(Built-in class PhaseOracle in Qiskit)

Quantum Minimum Finding finds a solution in time $\sim \sqrt{2^n}$

(n = number of variables)

Super-quadratic speedups

arXiv > quant-ph > arXiv:2212.01513

Sea

Help

Quantum Physics

[Submitted on 3 Dec 2022]

Mind the gap: Achieving a super-Grover quantum speedup by jumping to the end

Alexander M. Dalzell, Nicola Pancotti, Earl T. Campbell, Fernando G. S. L. Brandão

We present a quantum algorithm that has rigorous runtime guarantees for several families of binary optimization problems, including Quadratic Unconstrained Binary Optimization (QUBO), Ising spin glasses (p-spin model), and k-local constraint satisfaction problems (k-CSP). We show that either (a) the algorithm finds the optimal solution in time $O^*(2^{(0.5-c)n})$ for an n-independent constant c, a 2^{cn} advantage over Grover's algorithm; or (b) there are sufficiently many low-cost solutions such that classical random guessing produces a $(1-\eta)$ approximation to the optimal cost value in sub-exponential time for arbitrarily small choice of η. Additionally, we show that for a large fraction of random instances from the k-spin model and for any fully satisfiable or slightly frustrated k-CSP formula, statement (a) is the case. The algorithm and its analysis is largely inspired by Hastings' short-path algorithm [Quantum **2** (2018) 78].

For some hard binary optimization problems encoded into Hamiltonians:

Grover's algorithm can help traverse regions of small spectral gaps in the adiabatic evolution

Provable running time of $\sim \sqrt{2^{(1-c)n}}$

(for a small $c < 1$)

Finding the smallest element

x	$f(x)$
1	6
2	2
3	9
4	8
5	3
6	1
7	8
$N = 8$	4

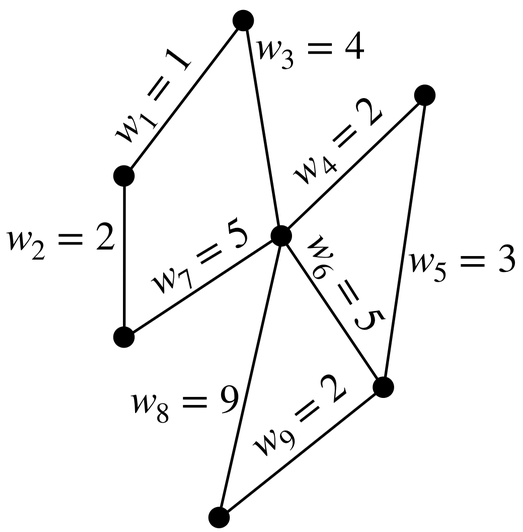
Quantum Minimum Finding can find the minimum in a search space of size N by using $\sim \sqrt{N}$ queries

As a standalone optimization algorithm, this is often ineffective since the size N of the search space is typically enormous

… but it can become useful when used as a subroutine

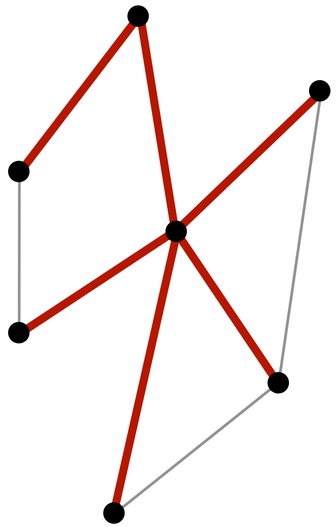
Example: Minimum Spanning Tree

Find a spanning tree with minimum total edge weight

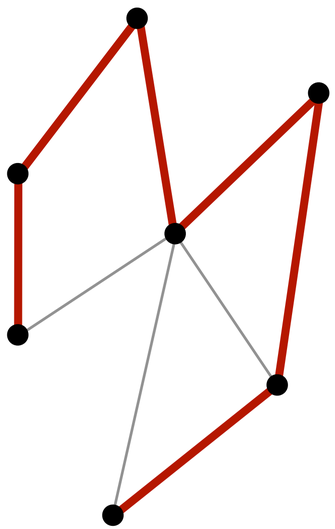


Number of vertices: $n = 7$

Number of edges: $m = 9$



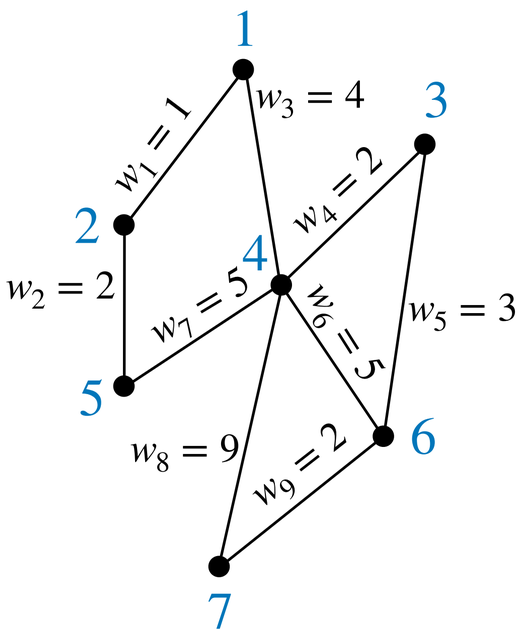
Weight: $w_1 + w_3 + w_4 + w_6 + w_7 + w_8 = 26$



Weight: $w_1 + w_2 + w_3 + w_4 + w_5 + w_9 = 14$

Minimum

Find a spanning tree with minimum
total edge weight



Number of vertices: $n = 7$

Number of edges: $m = 9$

Oracle

The oracle provides the vertices and weight of each edge

1	(1,2), 1
2	(2,5), 2
…	
9	(6,7), 2

Finding a minimum spanning tree in a graph
with n vertices and m edges

Number of classical queries

$$\sim m$$

Number of quantum queries

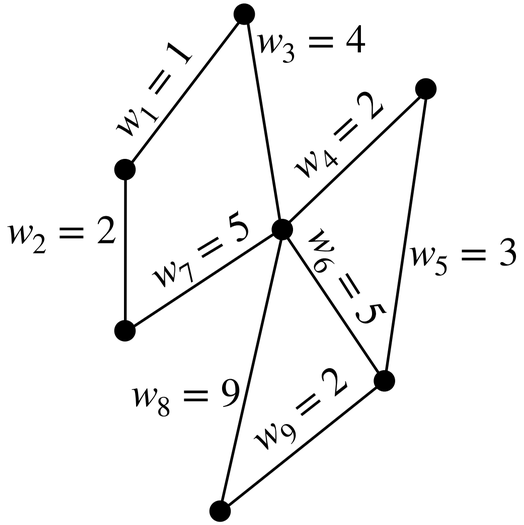
$$\sim \sqrt{nm}$$

$n - 1 \leq m \leq n^2$ if the graph is connected

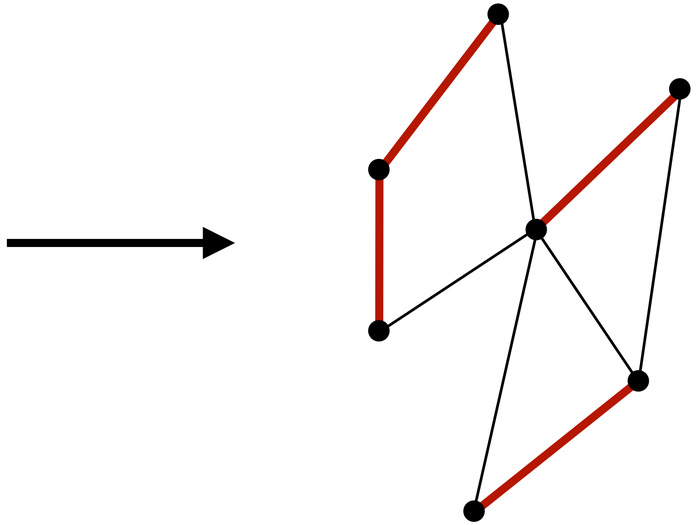
Borůvka's algorithm

Grow a spanning tree by adding the smallest-weight outgoing edge to each component of the current forest

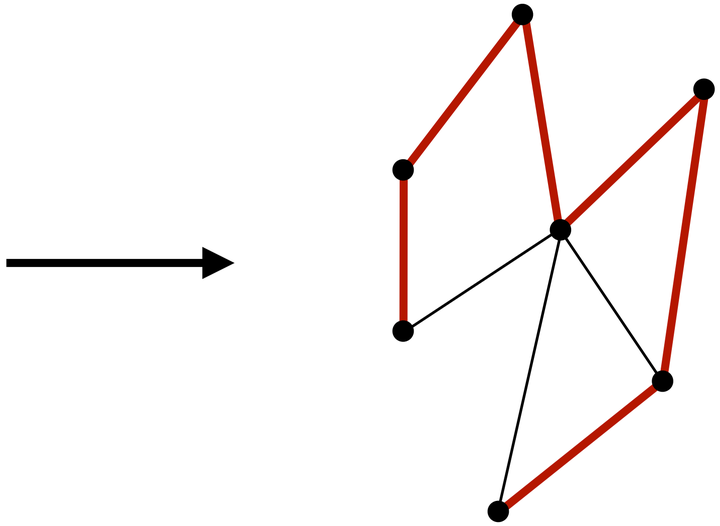
Step 0



Step 1



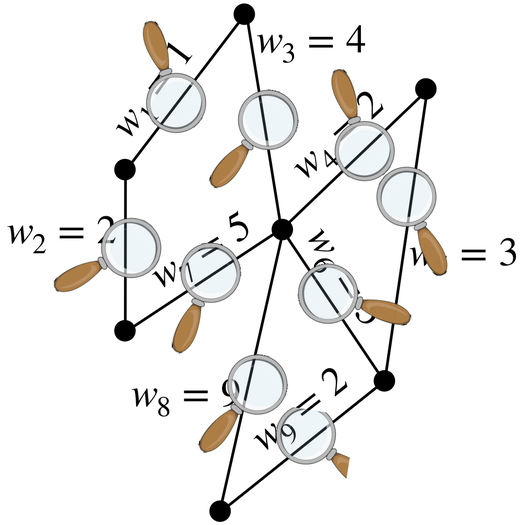
Step 2



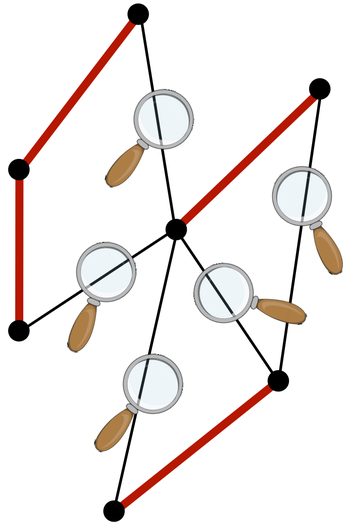
Borůvka's algorithm

Each step consists of finding the k-out-of-m edges of smallest weights that are outgoing from the k remaining trees.

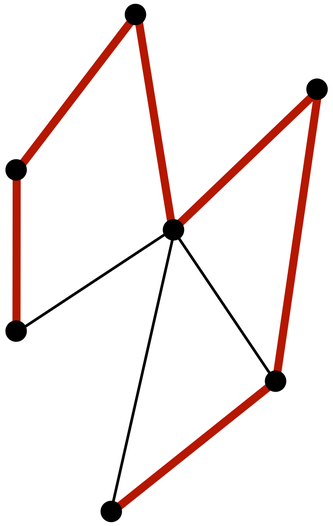
Step 0



Step 1



Step 2



Borůvka's algorithm

Each step consists of finding the k-out-of-m edges of smallest weights that are outgoing from the k remaining trees.

Quantum k-minimum finding

Quantum speedup (Dürr et al.'2006)

- Quantum minimum finding can find those edges using $\sim \sqrt{km}$ quantum queries
- At the t-th step of the algorithm, at most $n/2^t$ connected components remain

Overall complexity: $\sim \sqrt{nm} + \sqrt{nm/2} + \sqrt{nm/4} + \sqrt{nm/8} \sim \sqrt{nm}$

Other applications

arXiv > quant-ph > arXiv:quant-ph/0401091

Search...

Help | Adv

Quantum Physics

[Submitted on 15 Jan 2004 (v1), last revised 8 Jun 2004 (this version, v2)]

Quantum query complexity of some graph problems

Christoph Durr, Mark Heiligman, Peter Hoyer, Mehdi Mhalla

Quantum algorithms for graph problems are considered, both in the adjacency matrix model and in an adjacency list-like array model. We give almost tight lower and upper bounds for the bounded error quantum query complexity of Connectivity, Strong Connectivity, Minimum Spanning Tree, and Single Source Shortest Paths. For example we show that the query complexity of Minimum Spanning Tree is in Theta(n^{3/2}) in the matrix model and in Theta(sqrt{nm}) in the array model, while the complexity of Connectivity is also in Theta(n^{3/2}) in the matrix model, but in Theta(n) in the array model. The upper bounds utilize search procedures for finding minima of functions under various conditions.

Main tool: Minimum Finding

Examples of applications:

- Minimum Spanning Tree
- Single Source Shortest Paths

arXiv > quant-ph > arXiv:1911.07306

Search...

Help | Adv

Quantum Physics

[Submitted on 17 Nov 2019 (v1), last revised 8 May 2023 (this version, v4)]

Quantum Speedup for Graph Sparsification, Cut Approximation and Laplacian Solving

Simon Apers, Ronald de Wolf

Graph sparsification underlies a large number of algorithms, ranging from approximation algorithms for cut problems to solvers for linear systems in the graph Laplacian. In its strongest form, "spectral sparsification" reduces the number of edges to near-linear in the number of nodes, while approximately preserving the cut and spectral structure of the graph. In this work we demonstrate a polynomial quantum speedup for spectral sparsification and many of its applications. In particular, we give a quantum algorithm that, given a weighted graph with n nodes and m edges, outputs a classical description of an ϵ-spectral sparsifier in sublinear time $\tilde{O}(\sqrt{mn}/\epsilon)$. This contrasts with the optimal classical complexity $\tilde{O}(m)$. We also prove that our quantum algorithm is optimal up to polylog-factors. The algorithm builds on a string of existing results on sparsification, graph spanners, quantum algorithms for shortest paths, and efficient constructions for k-wise independent random strings. Our algorithm implies a quantum speedup for solving Laplacian systems and for approximating a range of cut problems such as min cut and sparsest cut.

Main tool: Graph sparsification

Examples of applications:

- Laplacian system solving
- Cut approximations

arXiv > quant-ph > arXiv:1807.05209

Search...

Help | Adva

Quantum Physics

[Submitted on 13 Jul 2018]

Quantum Speedups for Exponential-Time Dynamic Programming Algorithms

Andris Ambainis, Kaspars Balodis, Jānis Iraids, Martins Kokainis, Krišjānis Prūsis, Jevgēnijs Vihrovs

In this paper we study quantum algorithms for NP-complete problems whose best classical algorithm is an exponential time application of dynamic programming. We introduce the path in the hypercube problem that models many of these dynamic programming algorithms. In this problem we are asked whether there is a path from 0^n to 1^n in a given subgraph of the Boolean hypercube, where the edges are all directed from smaller to larger Hamming weight. We give a quantum algorithm that solves path in the hypercube in time $O^*(1.817^n)$. The technique combines Grover's search with computing a partial dynamic programming table. We use this approach to solve a variety of vertex ordering problems on graphs in the same time $O^*(1.817^n)$, and graph bandwidth in time $O^*(2.946^n)$. Then we use similar ideas to solve the travelling salesman problem and minimum set cover in time $O^*(1.728^n)$.

Main tool: Dynamic programming

Examples of applications:

- Travelling Salesman Problem
- Minimum Set Cover

arXiv > quant-ph > arXiv:1612.06203

Search...

Help | Adv

Quantum Physics

[Submitted on 19 Dec 2016]

Quantum speedup of the Travelling Salesman Problem for bounded-degree graphs

Alexandra E. Moylett, Noah Linden, Ashley Montanaro

The Travelling Salesman Problem is one of the most famous problems in graph theory. However, little is currently known about the extent to which quantum computers could speed up algorithms for the problem. In this paper, we prove a quadratic quantum speedup when the degree of each vertex is at most 3 by applying a quantum backtracking algorithm to a classical algorithm by Xiao and Nagamochi. We then use similar techniques to accelerate a classical algorithm for when the degree of each vertex is at most 4, before speeding up higher-degree graphs via reductions to these instances.

arXiv > cs > arXiv:1906.10375

Search...

Help | Adv

Computer Science > Data Structures and Algorithms

[Submitted on 25 Jun 2019]

Quantum speedup of branch-and-bound algorithms

Ashley Montanaro

Branch-and-bound is a widely used technique for solving combinatorial optimisation problems where one has access to two procedures: a branching procedure that splits a set of potential solutions into subsets, and a cost procedure that determines a lower bound on the cost of any solution in a given subset. Here we describe a quantum algorithm that can accelerate classical branch-and-bound algorithms near-quadratically in a very general setting. We show that the quantum algorithm can find exact ground states for most instances of the Sherrington-Kirkpatrick model in time $O(2^{0.226n})$, which is substantially more efficient than Grover's algorithm.

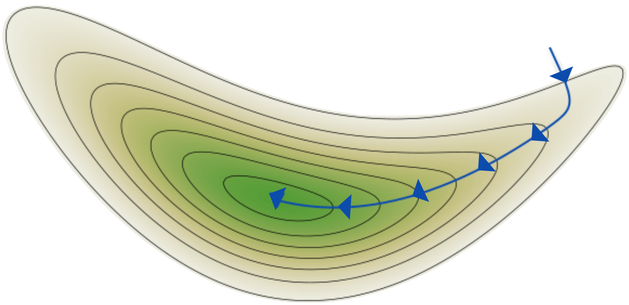
Main tools: Quantum walks, Backtracking, Branch-and-bound

Examples of applications:

- Travelling Salesman Problem
- Ground states of spin models

Gradient computation

Gradient descent is an optimization method that explores the search space by making iterative steps in the direction where f decreases the fastest



$$f : \mathbb{R}^d \to \mathbb{R}$$

This works very well in convex optimization (converges to the minimum) but is also used in non-convex optimization (converges to local minima)

The fastest decreasing direction is given by the gradient (~ derivative) of the objective function

Gradient

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_d}(x) \right)$$

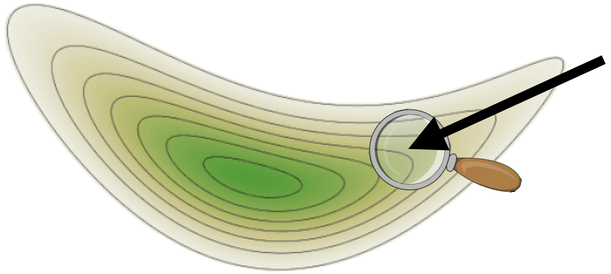
Gradient step

$$x^{(t+1)} = x^{(t)} - \nabla f(x^{(t)})$$

Gradient computation

$$f : \mathbb{R}^d \to \mathbb{R} \qquad \nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_d}(x) \right)$$

Suppose f behaves locally as a linear function:



$$f(x) = a_0 + a_1 x_1 + \ldots + a_d x_d$$

How many evaluations of f to compute its gradient $\nabla f(x) = (a_1, \ldots, a_d)$?

Number of classical queries: $d + 1$

Number of quantum queries: 2

$$\begin{cases} f(b_0) = a_0 + a_1 b_{0,1} + \ldots + a_d b_{0,d} \\ f(b_1) = a_0 + a_1 b_{1,1} + \ldots + a_d b_{1,d} \\ \quad \vdots \\ f(b_d) = a_0 + a_1 b_{d,1} + \ldots + a_d b_{d,d} \end{cases}$$

System of $d + 1$ independent equations with unique solution $(a_0, \ldots, a_d)$

Jordan's algorithm (2004)

Jordan's algorithm

Compute the gradient of a linear function: $f(x) = a_0 + a_1 x_1 + \ldots + a_d x_d$ $\quad f : \mathbb{R}^d \to \mathbb{R}$

$$\frac{1}{|G|^{1/2}} \sum_{x \in G} |x\rangle |0\rangle \longrightarrow \frac{1}{|G|^{1/2}} \sum_{x} e^{if(x)} |x\rangle |0\rangle \longrightarrow |a\rangle |0\rangle = |\nabla f(x)\rangle |0\rangle$$

Two quantum queries to U_f

Carefully chosen set of points $G \subset \mathbb{R}^d$

Quantum Fourier Transform

Reminiscent of the Bernstein–Vazirani problem: $f(x) = a_1 x_1 + \ldots + a_d x_d \bmod 2$ $\quad f : \{0,1\}^d \to \{0,1\}$

$$\frac{1}{2^{d/2}} \sum_{x \in \{0,1\}^d} |x\rangle |0\rangle \longrightarrow \frac{1}{2^{d/2}} \sum_{x} |x\rangle |f(x)\rangle \longrightarrow \frac{1}{2^{d/2}} \sum_{x} (-1)^{f(x)} |x\rangle |f(x)\rangle \longrightarrow \frac{1}{2^{d/2}} \sum_{x} (-1)^{f(x)} |x\rangle |0\rangle \longrightarrow |a\rangle |0\rangle$$

Quantum query U_f

Phase encoding (Z gates)

Inverse quantum query U_f^{-1}

Hadamard transform $H^{\otimes d}$

Jordan's algorithm

For general functions $f : \mathbb{R}^d \to \mathbb{R}$:

First-order approximation around current iterate:

$$f(x) \approx \overbrace{f(x^{(t)}) - \nabla f(x^{(t)})^\top \cdot x^{(t)}}^{a_0} + \overbrace{\nabla f(x^{(t)})_1}^{a_1} \cdot x_1 + \ldots + \overbrace{\nabla f(x^{(t)})_d}^{a_d} \cdot x_d$$

Under sufficient smoothness assumptions (f is analytic and has bounded partial derivatives):

Estimate $\left(\frac{\partial f}{\partial x_1}(x) \pm \epsilon, \ldots, \frac{\partial f}{\partial x_d}(x) \pm \epsilon\right)$ with $\sim \sqrt{d}/\epsilon$ quantum queries

Caveats:

- requires to evaluate f with high precision
- may not be competitive against non-oracular classical methods (ex: automatic differentiation)

Higher-order methods

Second-order methods: Use the Hessian matrix $\mathbf{H}_f$ (~ second-order derivative) as well

- ✓ Faster convergence rate compared to gradient descent
- ✗ Harder to compute and requires more memory (matrix of size d^2)

Example of application: interior point methods

Quantum speedups investigated by arXiv:1808.09266, arXiv:2311.03215, …

Monte-Carlo algorithms

Monte-Carlo algorithms rely on random sampling to make their decisions

(ex: stochastic optimization)

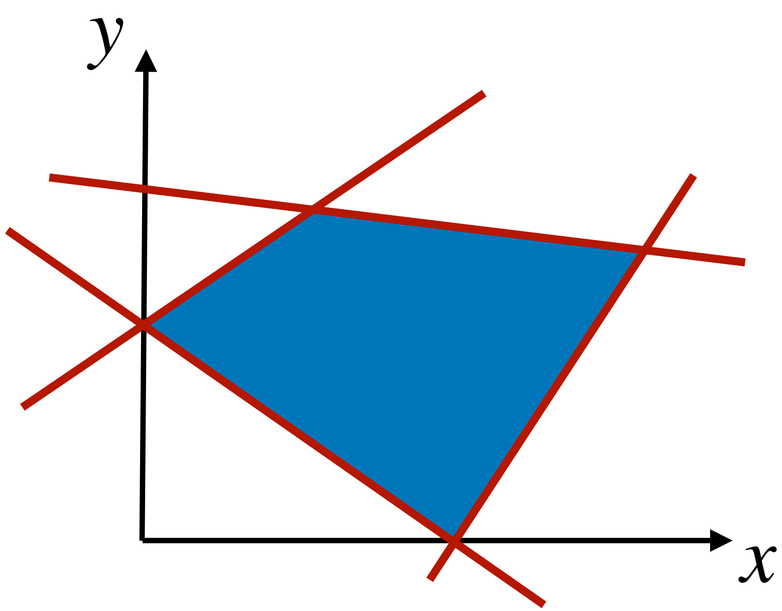
Quantum computers can sample certain distributions more efficiently

than classical computers

Linear programming

Linear programming

Minimize a real-valued linear objective function subject to linear constraints



Convex optimization problem

$$\text{Minimize } 2x + y$$

$$\begin{aligned} \text{s.t.} \quad & y \leq x + 2 \\ & y \geq 3x - 4 \\ & 4y \leq -x + 12 \\ & 2y \geq -3x + 4 \\ & x, y \geq 0 \end{aligned}$$

$n = 2$ variables

$m = 4$ constraints

The coefficients of the LP are provided via an oracle

Find an ϵ-approximate solution?

Quantum solver

$\sim \sqrt{n+m}/\epsilon^{2.5}$

Classical solvers

At least $n + m$

Grigoriadis-Khachiyan's algorithm

Linear programming can be reduced to a problem where:

- The linear constraints are arranged into a skew-symmetric matrix: $A \in [-1,1]^{N \times N}$

$$A = -A^\top$$

- The search space is the set of all probability vectors: $\Omega = \left\{ x \in [0,1]^N, \ \sum_i x_i = 1 \right\}$

- The goal is to find an $x \in \Omega$ such that: $(Ax)_i \leq \epsilon, \forall i$

(Nash equilibrium: there exists $x^\star \in \Omega$ such that $Ax^\star = (0,\ldots,0)$)

Grigoriadis-Khachiyan's algorithm

Find proba. vector $x \in [0,1]^N$, $\sum_i x_i = 1$

satisfying $(Ax)_i \leq \epsilon, \forall i$

where the entries $A \in [-1,1]^{N \times N}$
are provided via an oracle

Ansatz: Gibbs distribution

$$x \propto e^{\epsilon A u}$$

where $u \in \mathbb{N}^N$ is an integer-valued vector

Set $t = 0$ and $u^{(0)} = (0,\ldots,0)$ $\qquad x^{(0)} = (1/N, \ldots, 1/N)$

Repeat:

Sample $i \in [N]$ from the Gibbs distribution $x^{(t)} \propto e^{\epsilon A u^{(t)}}$

Increment the i-th coordinate: $u^{(t+1)} = u^{(t)} + e_i$

Increment the time step: $t = t + 1$

The distribution leans toward
the unsatisfied constraints

Converges to a solution
in $t \sim \log(N)/\epsilon^2$ steps,
each of cost $\sim N$

Quantum algorithm

Find proba. vector $x \in [0,1]^N$, $\sum_i x_i = 1$

satisfying $(Ax)_i \leq \epsilon, \forall i$

where the entries $A \in [-1,1]^{N \times N}$
are provided via an oracle

Ansatz: Gibbs distribution

$$x \propto e^{\epsilon A u}$$

where $u \in \mathbb{N}^N$ is an integer-valued vector

Set $t = 0$ and $u^{(0)} = (0,\ldots,0)$ $\qquad x^{(0)} = (1/N, \ldots, 1/N)$

Repeat:

Sample $i \in [N]$ from the Gibbs distribution $x^{(t)} \propto e^{\epsilon A u^{(t)}}$

Increment the i-th coordinate: $u^{(t+1)} = u^{(t)} + e_i$

Increment the time step: $t = t + 1$

Quantum speedup:

Prepare and measure: $\propto \sum_{i=1}^{N} e^{\epsilon A_i u^{(t)}/2} |i\rangle$

Cost per step: $\sim \sqrt{N}$ queries

Quantum algorithm

Find proba. vector $x \in [0,1]^N$, $\sum_i x_i = 1$

satisfying $(Ax)_i \leq \epsilon, \forall i$

where the entries $A \in [-1,1]^{N \times N}$
are provided via an oracle

Ansatz: Gibbs distribution

$$x \propto e^{\epsilon A u}$$

where $u \in \mathbb{N}^N$ is an integer-valued vector

Quantum state preparation of $\propto \sum_{i=1}^{N} e^{\epsilon A_i u^{(t)}/2} |i\rangle$:

Assuming $\max_i A_i u^{(t)} = 0$

Prepare the uniform superposition $\longrightarrow$ Apply a block-encoding U of matrix $e^{\epsilon A u^{(t)}/2}$ $\longrightarrow$ Amplify the first part

$$\frac{1}{\sqrt{N}} \sum_i |i\rangle \qquad U\left(\frac{1}{\sqrt{N}} \sum_i |i\rangle |0\rangle\right) = \left(\frac{1}{\sqrt{N}} \sum_i e^{\epsilon A_i u^{(t)}/2} |i\rangle\right) |0\rangle + |\ldots\rangle |0^{\perp}\rangle \qquad \frac{1}{\ldots} \sum_i e^{\epsilon A_i u^{(t)}/2} |i\rangle |0\rangle$$

Quantum algorithm

arXiv > quant-ph > arXiv:1904.03180

Search...

Help | Adv

Quantum Physics

[Submitted on 5 Apr 2019]

Quantum algorithms for zero-sum games

Joran van Apeldoorn, András Gilyén

We derive sublinear-time quantum algorithms for computing the Nash equilibrium of two-player zero-sum games, based on efficient Gibbs sampling methods. We are able to achieve speed-ups for both dense and sparse payoff matrices at the cost of a mildly increased dependence on the additive error compared to classical algorithms. In particular we can find ε-approximate Nash equilibrium strategies in complexity $\tilde{O}(\sqrt{n+m}/\varepsilon^3)$ and $\tilde{O}(\sqrt{s}/\varepsilon^{3.5})$ respectively, where $n \times m$ is the size of the matrix describing the game and s is its sparsity. Our algorithms use the LP formulation of the problem and apply techniques developed in recent works on quantum SDP-solvers. We also show how to reduce general LP-solving to zero-sum games, resulting in quantum LP-solvers that have complexities $\tilde{O}(\sqrt{n+m}\gamma^3)$ and $\tilde{O}(\sqrt{s}\gamma^{3.5})$ for the dense and sparse access models respectively, where γ is the relevant "scale-invariant" precision parameter

arXiv > quant-ph > arXiv:2301.03763

Search...

Help | Adv

Quantum Physics

[Submitted on 10 Jan 2023]

Quantum Speedups for Zero-Sum Games via Improved Dynamic Gibbs Sampling

Adam Bouland, Yosheb Getachew, Yujia Jin, Aaron Sidford, Kevin Tian

We give a quantum algorithm for computing an ϵ-approximate Nash equilibrium of a zero-sum game in a $m \times n$ payoff matrix with bounded entries. Given a standard quantum oracle for accessing the payoff matrix our algorithm runs in time $\widetilde{O}(\sqrt{m+n} \cdot \epsilon^{-2.5} + \epsilon^{-3})$ and outputs a classical representation of the ϵ-approximate Nash equilibrium. This improves upon the best prior quantum runtime of $\widetilde{O}(\sqrt{m+n} \cdot \epsilon^{-3})$ obtained by [vAG19] and the classic $\widetilde{O}((m+n) \cdot \epsilon^{-2})$ runtime due to [GK95] whenever $\epsilon = \Omega((m+n)^{-1})$. We obtain this result by designing new quantum data structures for efficiently sampling from a slowly-changing Gibbs distribution.

Semidefinite programming

A similar (more involved) quantum algorithm applies to solving semidefinite programs

$$
\begin{aligned}
&\text{Minimize } \operatorname{Tr}(CX) \\
&\text{subject to } \quad X \in \mathbb{R}^{2^n} \text{ is positive semidefinite} \\
&\qquad\qquad\quad \operatorname{Tr}(A_1 X) \leq b_1 \\
&\qquad\qquad\quad \operatorname{Tr}(A_2 X) \leq b_2 \\
&\qquad\qquad\quad \ldots \\
&\text{given } \quad C, A_1, A_2, \ldots \in \mathbb{R}^{2^n} \\
&\qquad\qquad\quad b_1, b_2, \ldots \in \mathbb{R}
\end{aligned}
$$

Semidefinite programming

A similar (more involved) quantum algorithm applies to solving semidefinite programs

The core ingredient is Quantum Gibbs Sampling:

Prepare the density matrix (proportional to) $e^{\epsilon H^{(t)}}$ for a certain Hamiltonian $H^{(t)}$

Caveats of such LP/SDP quantum solvers:

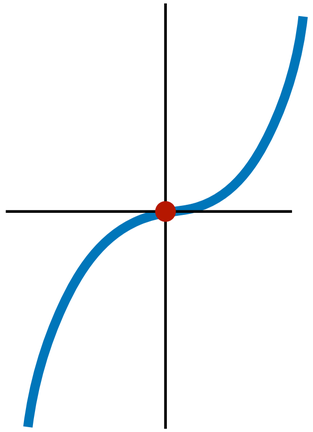
- poor scaling with precision ϵ
- involved quantum circuits for arithmetic operations
- may perform worse than other classical algorithms on LP/SDP of interest

Escaping Saddle Points

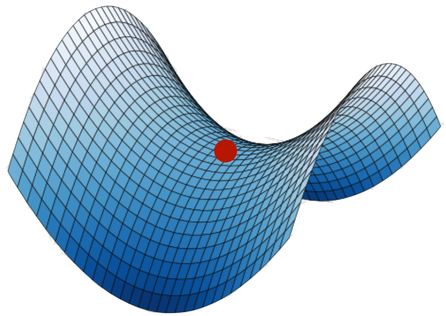
Saddle points

In continuous optimization, $x \in \mathbb{R}^d$ is a critical point if the gradient is zero $\nabla f(x) = 0$

- Gradient descent stops progressing in this situation
- If the function is non-convex, critical points can be local minima, local maxima, or saddle points



$y = x^3$



$z = x^2 - y^2$

Techniques for escaping saddle points:

- Compute the Hessian (expansive)
- Add noise (random perturbation) to current position

Quantum algorithm

arXiv:2007.10253

Move into a random position obtained by solving the Schrödinger equation:

$$i\frac{\partial}{\partial t}\psi(t,x) = \left(-\frac{1}{2}\Delta + f(x)\right)\psi(t,x)$$

Kinetic operator — Potential operator

Under the initial condition:

$$\psi(0,x) \propto \exp(-\|x^{(t)}-x\|^2/\sigma^2)$$

Isotropic Gaussian distribution centered at current saddle point

Quantum algorithm: Prepare $|\psi(t)\rangle$ with Hamiltonian simulation and measure it to get $x^{(t+1)}$

Cost: improved scaling with dimension d over classical methods

Quantum algorithm

arXiv:2007.10253

Move into a random position obtained by solving the Schrödinger equation:

$$i\frac{\partial}{\partial t}\psi(t,x) = \left(-\frac{1}{2}\Delta + f(x)\right)\psi(t,x)$$

Kinetic operator — Potential operator

Under the initial condition:

$$\psi(0,x) \propto \exp(-\|x^{(t)} - x\|^2/\sigma^2)$$

Isotropic Gaussian distribution centered at current saddle point

Why it works: Quadratic approximation of f at $x^{(t)}$:

$$f(x) \approx f(x^{(t)}) + \cancel{\nabla f(x^{(t)})^\top (x - x^{(t)})} + \frac{1}{2}(x - x^{(t)})^\top \cdot \mathbf{H}_f(x^t) \cdot (x - x^{(t)})$$

$\Rightarrow \psi(t,x)$ follows a multivariate Gaussian $\mathcal{N}(x^{(t)}, \Sigma(t))$ that drifts toward the negative curvature region of f over time