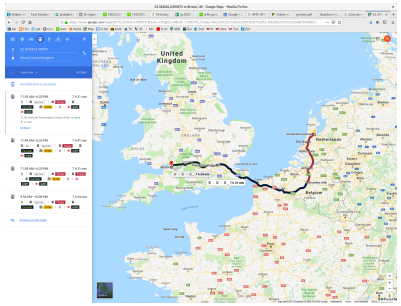
Quantum Algorithms for Optimization

Ronald de Wolf

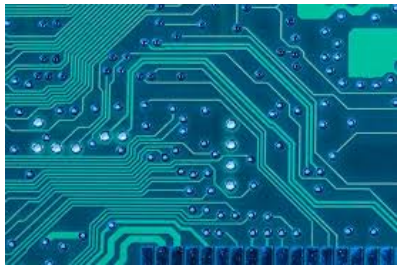
CWI QuSoft Universiteit van Amsterdam

Optimization

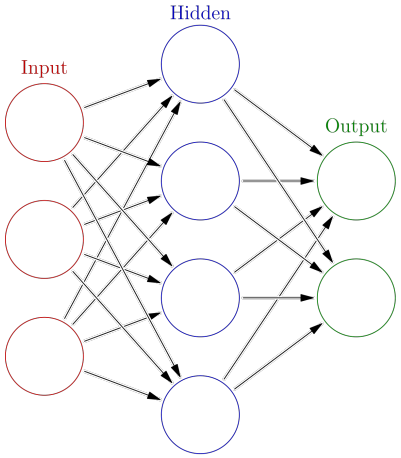
General problem: $\min_{x \in K} f(x)$



- Finding the shortest route on a given map



- Designing a more energy-efficient chip



- Training your neural network to detect cats

Discrete and continuous settings

Combinatorial optimization: variables are discrete (bits, integers):

- Shortest path algorithms
- Matching algorithms
- Max flow / Min cut in a network
- Often discrete optimization problems are NP-hard: Constraint-satisfaction, Traveling Salesman, protein folding, ...

Continuous optimization: variables are continuous (reals):

- Gradient descent
- Linear programs, semidefinite programs
- Non-convex optimization

Or a mix of these

How can quantum computers help?

Faster optimization is one of the main potential application areas of quantum computers (along with cryptography and simulation). Several new quantum algorithms discovered in the last 5 years

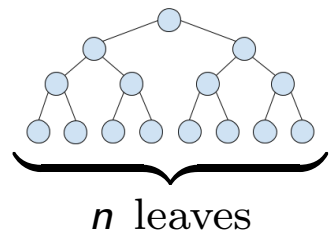
The goal of this talk: **survey what we know**

Warning about what we need to assume:

- Should be able to evaluate function on superposition of inputs
- If we are given classical data (eg, input graph, or data for learning) we should be able to access this in superposition.

  Accessing classical n-bit RAM takes $\log n$ steps

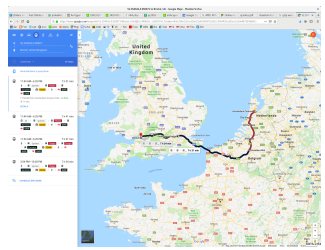
  Quantum RAM should be the same;
  but hard to implement with noise



Quantum speed-up for discrete optimization

- Find the minimum of $f : \{1, \ldots, n\} \to \mathbb{R}$
  in $\sim \sqrt{n}$ f-evaluations and other operations (Dürr-Høyer'96)

- Finding shortest path in an n-vertex graph
  classical complexity of $O(n^2)$ (Dijkstra'56)
  vs quantum complexity $O(n^{1.5})$ (Dürr et al.'04)



- Polynomial speedups for matching, max flows

- These typically use Grover's quantum search
  as a blackbox within larger classical algorithm

Quantum speedup for NP-hard optimization problems

We don't expect quantum computers to have exponential speedup for NP-hard problems. But polynomial speedups are possible:

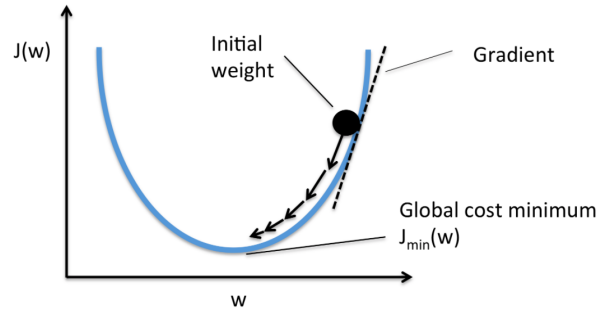
- Find a satisfying assignment to a formula ϕ on n Boolean variables $x_1, \ldots, x_n$ in $\sim \sqrt{2^n}$ steps using Grover
- If ϕ is a 3-SAT formula, then plain Grover is slower than classical Schöning algorithm, which takes time $\sim (4/3)^n$. Can be quadratically improved with amplitude amplification

Two other methods for quantum speed-up:

- Montanaro'15: quadratic speedup for backtracking
- Ambainis et al.'18: small polynomial speedups for some dynamic programming algorithms, incl. TSP ($2^n \to 1.7^n$)

Quantum speedup for gradient descent

- Gradient descent: iterative method to find local minimum of f on $\mathbb{R}^n$



1. Start with $t = 0$, and some initial point $x^{(0)}$
2. Compute the gradient $\nabla f = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})$ in $x^{(t)}$
3. Move down for some stepsize: $x^{(t+1)} \leftarrow x^{(t)} - \eta \cdot \nabla f(x^{(t)})$
4. Set $t \leftarrow t + 1$, goto 2

- Used a lot, studied a lot, often fast convergence
- Quantum computers can speed this up in some cases by computing gradient more efficiently

Computing gradient in given point z

- Jordan'04: assume f is approximately linear around z: $f(x) \approx a_0 + \sum_{j=1}^{n} a_j x_j$. NB: $\nabla f(z) \approx (a_1, \ldots, a_n)$
- Suppose we can compute f "in the phase": $|x\rangle \to e^{if(x)}|x\rangle$

  Create superposition, x_j ranges over finite grid G around z_j:

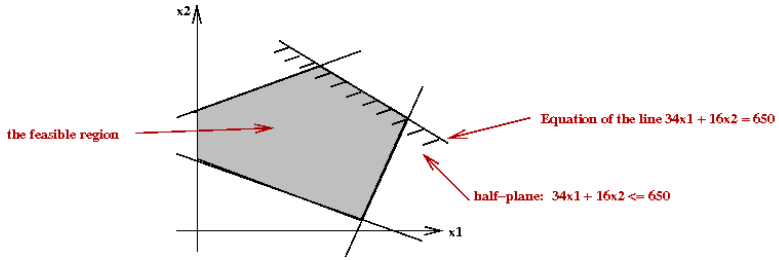
$$\frac{1}{\sqrt{|G|^n}} \sum_{x \in G^n} e^{if(x)}|x\rangle \approx e^{ia_0} \bigotimes_{j=1}^{n} \frac{1}{\sqrt{|G|}} \sum_{x_j \in G} e^{ia_j x_j}|x_j\rangle.$$

  Applying n-fold quantum Fourier transform gives $a_1, \ldots, a_n$
- Issue: may need to take G very close to z before $f \approx$ linear
  This requires computing f with very high precision
- Improved by Gilyén, Arunachalam, Wiebe'18 for case of mildly-smooth f, using higher-level interpolation formulas

Quantum speedup for LPs and SDPs

- Linear program:
  max/min linear function of $x \in \mathbb{R}^n$
  subject to m linear constraints

$$\begin{array}{ll} \max & x_1 + 4x_2 \\ \text{s.t.} & 34x_1 + 16x_2 \leq 650 \\ & \cdots \qquad\quad \leq 100 \\ & \cdots \\ & x_1, x_2 \geq 0 \end{array}$$



- Semidefinite program: replace x by psd matrix $X \in \mathbb{R}^{n \times n}$
- LP and SDP are solvable classically in polynomial time
- Brandão-Svore'16: can speed up classical "primal-dual" SDP-solvers by treating X as a $\log(n)$-qubit state
- van Apeldoorn-Gilyén'18: current best algorithm for s-sparse SDPs: $O((\sqrt{m}+\sqrt{n})s\gamma^5)$; γ related to desired error

Other quantum methods for optimization

- Harrow-Hassidim-Lloyd'08 algorithm "solves" large, well-conditioned linear systems. No convincing applications with exponential speedup and reasonable input-assumptions yet
- Polynomial quantum speedup for finite-element methods (Montanaro-Pallister'15)
- Fast reductions between membership and separation oracles for convex optimization (AGGW'18, CCLW'18)
- Quantum interior point method (Kerenidis-Prakash'18)
- Heuristics: (simulated) quantum annealing,
  QAOA algorithm (Farhi-Goldstone-Gutmann'14).
  Shallow quantum circuit parametrized by few parameters.
  Run it, measure the output, adjust parameters to improve.

Summary

- Faster optimization is
  one of the main potential applications of quantum computers
- We have many quantum speedups, usually polynomial:
  - Discrete: minimizing over a finite set, shortest path
  - Continuous: gradient descent, linear/semidefinite programs
- In the NISQ era (Noisy Intermediate-Scale Quantum):
  we could start to experiment with heuristics like QAOA