Quantum Speedup of the Travelling Salesman Problem on Bounded Degree Graphs

Dominic J. Moylett[1]

School of Physics and Department of Electrical and Electronic Engineering
University of Bristol

dominic.moylett@bristol.ac.uk

October 19, 2016

[1]Work completed in collaboration with Ashley Montanaro and Noah Linden.

Based on a True Story

I graduated in June 2015.
To celebrate, I wanted to do a tour of Europe.

Interrail

2

[2] http://www.interrail.eu/plan-your-trip/interrail-railway-map

Where we wanted to visit

What's the fastest way of visiting every location?

The Travelling Salesman Problem

Given a graph $G = (V, E)$ with cost matrix C, what is Hamiltonian cycle with the lowest cost?
This problem is NP-hard, so finding a polynomial time solution would prove that $P = NP$.
A naïve solution would take $O(|V|!)$ time by searching over every permutation of the vertices.
Grover search could be applied to this approach to achieve $O(\sqrt{|V|!})$ time.
But other more efficient classical algorithms exist. Can these be sped up using quantum algorithms?

TODO: Add graph.

Our Results

- A quadratic speedup for the Travelling Salesman Problem when the degree of any vertex in the graph is at most 3.
- This is using a classical algorithm by Eppstein[3], combined with a quantum speedup by Montanaro[4].
- Speedups for graphs of higher bounded degrees are found by reducing to this case.

[3] D. Eppstein, *Journal of Graph Algorithms and Applications*, **11**(1), pp. 61–81 (2007)
[4] A. Montanaro, *arXiv:1509.02374*

The Difficulty of a Quantum Speedup

Key components for many algorithms have limited interesting quantum speedup:

- The Held-Karp algorithm[5] relies on dynamic programming (no known speedup).
- Christofides' algorithm[6] relies on finding a minimum weight perfect matching (speedup for bipartite graphs).
- Algorithms which use Nearest Neighbour can be sped up from $O(n^2)$ to $O(n^{3/2})$ time via Grover search.

Others algorithms do not even have known performance[7].

[5] M. Held and R. M. Karp, *Journal of the Society for Industrial and Applied Mathematics*, **10**(1), pp. 196-210 (1962)

[6] N. Christofides, *Management Sciences Research Report 388, Graduate School of Industrial Administration, CMU* (1976)

[7] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, *Science*, **220**(4598), pp. 671–680 (1983)

Eppstein's Algorithm

Solves the Travelling Salesman Problem on graphs where the degree of any vertex is at most 3.

Takes as input a graph $G = (V, E)$ and a set $F \subseteq E$ of "forced" edges.

Returns the shortest Hamiltonian cycle which includes every edge in F.

Main steps:

- Reduce (G, F) to (G', F') by changing specific subgraphs in G.
- If F' contains a Hamiltonian cycle then return then length of F'.
- If a Hamiltonian cycle cannot be made, then abort.
- Pick an edge $xy \in V' \setminus F'$ according to specific rules.
- Call Eppstein's algorithm on $(G', F' \cup \{xy\})$ and $(G' \setminus \{xy\}, F')$.
- Return the Hamiltonian cycle length with the shortest path. If both branches abort then abort.

Edge Picking

The main bottleneck of Eppstein's algorithm is the recursive calls. So how is the edge to force or remove chosen?

1. If $G' \setminus F'$ has a cycle of four vertices such that two of the vertices are incident to forced edges, pick one vertex x in the cycle that is not incident to a forced edge and another vertex y not in the cycle.
2. Otherwise if $F' \neq \emptyset$, pick an edge $(y, z) \in F$ and then chooses an adjacent edge $(x, y) \notin F$ as long as (x, y) is not part of a disjoint cycle of four vertices in $G \setminus F$.
3. Otherwise pick any edge.

Performance

Eppstein showed that these branching rules broke the problem size into one of two cases:

1. A subproblem with two more forced edges and one more cycle of 4 unforced edges, and a subproblem with three fewer vertices.
2. Two subproblems with three more forced edges.
3. A subproblem with two more forced edges, and a subproblem with five more forced edges.

Linear recurrence solving shows that the overall runtime of is $O(2^{n/3})$.

Why not use Grover search?

Grover search can be used if we can map a binary string to each result of the algorithm.

We can apply Grover search to Eppstein's algorithm by removing an edge if the next bit is 0 and forcing it if the next bit is 1.

However, the longest bit string we might need is $n/2$ bits, because of the case which only which only adds two forced edges to the graph.

Thus the best speedup we would see of Grover search is $O(2^{n/4})$ time.

We need to take advantage of the problem's structure if we want to do even better.

Backtracking Algorithms

Backtracking algorithms are algorithms to solve constraint satisfaction problems, where we have n variables to assign values to such that they satisfy m constraints.

They work by using a predicate P and heuristic h as follows, given a partial assigment V:

1. If $P(V) = \perp$ then abort.
2. Let $v_i \leftarrow h(V)$.
3. For all possible assignments a to v_i:
   1. Call recursively on $(V \cup (v_i, a))$
   2. If call does not abort then return $(V \cup (v_i, a))$.
4. Abort.

Quantum Speedup for Backtracking Algorithms

Montanaro proved the following:

Theorem (Montanaro)

Let T be the number of recursive calls made by a backtracking algorithm. For any $0 < \delta < 1$, there is a quantum algorithm which evaluates P and h $O(\sqrt{T}n^{3/2}\log(n)\log(1/\delta))$ times, and outputs a partial assignment V such that $P(V)$ is true or aborts if no such V exists. The algorithm fails with probability δ.

This speedup works by visualising the backtracking algorithm's calls as a tree and performing a quantum walk on the tree.

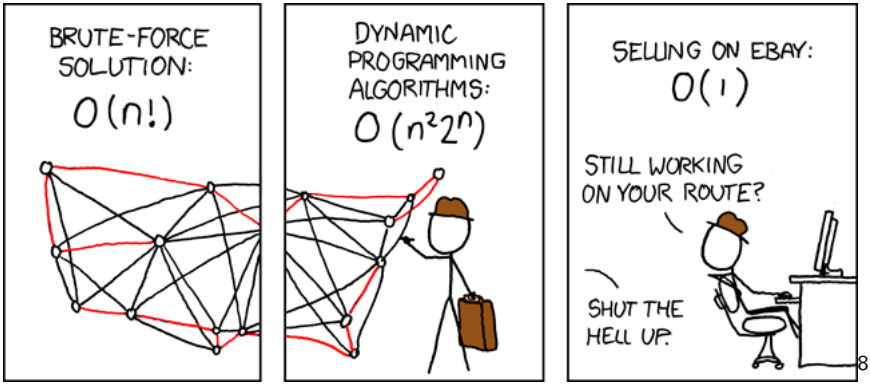
Conclusion

Future Steps:

- Apply further analysis to algorithms by Xiao and Nagamochi to see if they can perform even better.
- See if we can outperform Bjorklund et al. for bounded degree 7.
- ???
- Publish!

The End

Any questions?



[8]

[8] https://xkcd.com/399/

Post-credits

This is not the slide you're looking for.