Applying quantum algorithms to constraint satisfaction problems

Ashley Montanaro

Joint work with Earl Campbell (University of Sheffield) and Ankur Khurana (University of Bristol)

School of Mathematics,
University of Bristol

26 November 2018

arXiv:1810.05582

University of Bristol

EPSRC
Engineering and Physical Sciences Research Council

Introduction

There are quite a lot of quantum algorithms …

Introduction

There are quite a lot of quantum algorithms …

…but few cases where a substantial speedup has been calculated in detail for a practically relevant problem.

Introduction

There are quite a lot of quantum algorithms …

…but few cases where a substantial speedup has been calculated in detail for a practically relevant problem.

Some fully worked-out applications with large speedups:

- Nitrogen fixation [Reiher et al '17]
- Many-body localisation [Childs et al '17]
- Other problems in quantum chemistry and condensed-matter physics, e.g. [Babbush et al '18]
- Integer factorisation [Kutin '06]

Introduction

There are quite a lot of quantum algorithms …

…but few cases where a substantial speedup has been calculated in detail for a practically relevant problem.

Some fully worked-out applications with large speedups:

- Nitrogen fixation [Reiher et al '17]
- Many-body localisation [Childs et al '17]
- Other problems in quantum chemistry and condensed-matter physics, e.g. [Babbush et al '18]
- Integer factorisation [Kutin '06]

But what if we don't care about cryptography or simulation of quantum systems?

General applications

Question

Can we find a truly general-interest application of quantum computers?

General applications

Question

Can we find a truly general-interest application of quantum computers?

Desiderata:

1. There should be a quantum algorithm with provable correctness and performance bounds.

General applications

Question

Can we find a truly general-interest application of quantum computers?

Desiderata:

1. There should be a quantum algorithm with provable correctness and performance bounds.
2. It should solve a problem that (many) people care about in a reasonable time (e.g. < 1 day).

General applications

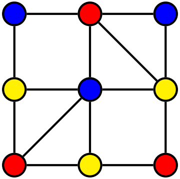
Question

Can we find a truly general-interest application of quantum computers?

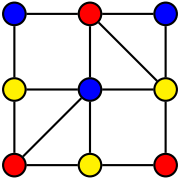
Desiderata:

1. There should be a quantum algorithm with provable correctness and performance bounds.
2. It should solve a problem that (many) people care about in a reasonable time (e.g. < 1 day).
3. We should compare it against the best classical algorithms running on real hardware.

Two applications: k-SAT and graph k-colouring

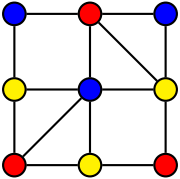


Two applications: k-SAT and graph k-colouring



$$(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$$

Two applications: k-SAT and graph k-colouring



$$(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$$

Each problem is NP-complete and has a huge number of direct applications:

- SAT: verification of electronic circuits; planning; computer-aided mathematical proofs; ...
- Colouring: register allocation; scheduling; frequency assignment problems; ...

Our results

1. We applied quantum algorithms (Grover's algorithm and a quantum approach for accelerating backtracking algorithms [AM '18]) to SAT and graph colouring.

Our results

1. We applied quantum algorithms (Grover's algorithm and a quantum approach for accelerating backtracking algorithms [AM '18]) to SAT and graph colouring.

2. We optimised the time complexity (circuit depth) of the algorithms.

Our results

1. We applied quantum algorithms (Grover's algorithm and a quantum approach for accelerating backtracking algorithms [AM '18]) to SAT and graph colouring.

2. We optimised the time complexity (circuit depth) of the algorithms.

3. We estimated their likely runtimes when applied to random instances.

Our results

1. We applied quantum algorithms (Grover's algorithm and a quantum approach for accelerating backtracking algorithms [AM '18]) to SAT and graph colouring.

2. We optimised the time complexity (circuit depth) of the algorithms.

3. We estimated their likely runtimes when applied to random instances.

4. We calculated the actual runtimes and other complexity measures, for various hardware parameter regimes.

Our results

1. We applied quantum algorithms (Grover's algorithm and a quantum approach for accelerating backtracking algorithms [AM '18]) to SAT and graph colouring.

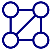
2. We optimised the time complexity (circuit depth) of the algorithms.

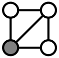
3. We estimated their likely runtimes when applied to random instances.

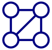
4. We calculated the actual runtimes and other complexity measures, for various hardware parameter regimes.

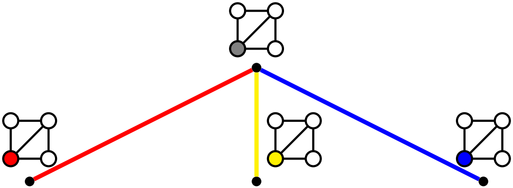
5. We compared against the likely performance of leading classical algorithms (Maple_LCM_Dist and DSATUR).

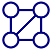
Colouring by backtracking

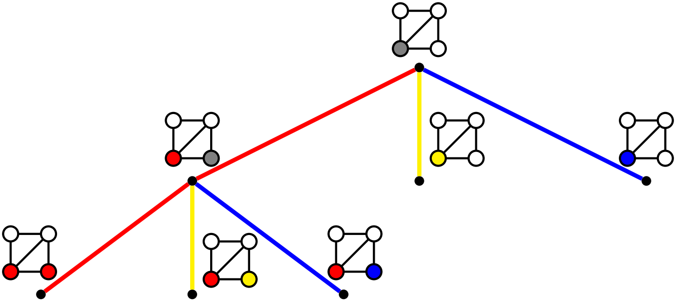
Colouring  by backtracking

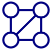


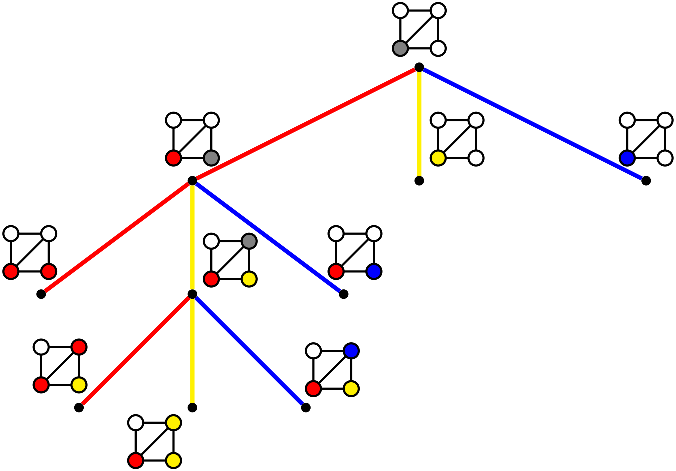
Colouring  by backtracking

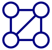


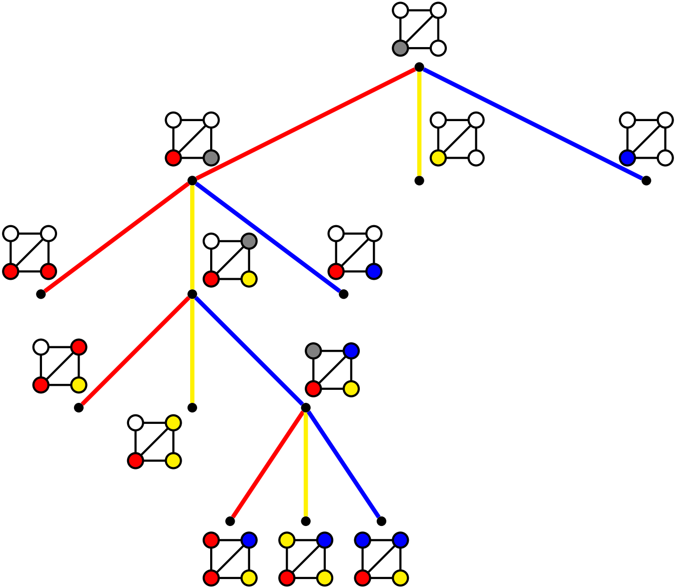
Colouring  by backtracking



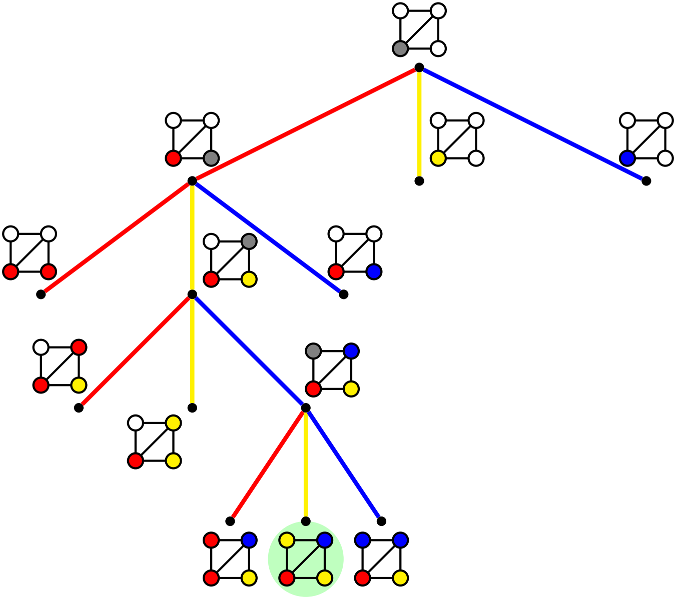
Colouring  by backtracking



Colouring  by backtracking



Colouring by backtracking



Cost model

We work out the runtime and space usage of quantum algorithms based on the use of the surface code [Fowler et al '12] for quantum error-correction.

Cost model

We work out the runtime and space usage of quantum algorithms based on the use of the surface code [Fowler et al '12] for quantum error-correction.

We then convert this to real-world runtimes based on various regimes corresponding to different parameters for quantum-computing hardware:

Parameter	Realistic	Plausible	Optimistic
Measurement time	50ns	5ns	0.5ns
2-qubit gate time	30ns	3ns	0.3ns
Gate error rate	10^{-3}	10^{-4}	10^{-5}

Cost model

We work out the runtime and space usage of quantum algorithms based on the use of the surface code [Fowler et al '12] for quantum error-correction.

We then convert this to real-world runtimes based on various regimes corresponding to different parameters for quantum-computing hardware:

Parameter	Realistic	Plausible	Optimistic
Measurement time	50ns	5ns	0.5ns
2-qubit gate time	30ns	3ns	0.3ns
Gate error rate	10^{-3}	10^{-4}	10^{-5}

"Realistic" is (approximately) achievable today; other two columns represent order-of-magnitude improvements.

Cost model

- To understand the cost of implementing a quantum circuit fault-tolerantly, it helps to split it into Clifford gates (easy) and T gates or Toffoli gates (hard).

Cost model

- To understand the cost of implementing a quantum circuit fault-tolerantly, it helps to split it into Clifford gates (easy) and T gates or Toffoli gates (hard).

- We assume that Clifford gates are free and T/Toffoli gates can be implemented in time equal to the cost of one measurement.

Cost model

- To understand the cost of implementing a quantum circuit fault-tolerantly, it helps to split it into Clifford gates (easy) and T gates or Toffoli gates (hard).

- We assume that Clifford gates are free and T/Toffoli gates can be implemented in time equal to the cost of one measurement.

- This is justified by the use of time-optimal techniques [Fowler '12, Eastin '13, ...] to prepare magic states offline and then inject them at the cost of 1 measurement.

Summary of results: good and bad news

- In the most optimistic hardware parameter regime, we could see speedup factors of $> 10^5$ (compared with a standard desktop PC) for k-SAT (via Grover's algorithm) and $> 10^4$ for graph colouring (via backtracking).

Summary of results: good and bad news

- In the most optimistic hardware parameter regime, we could see speedup factors of $> 10^5$ (compared with a standard desktop PC) for k-SAT (via Grover's algorithm) and $> 10^4$ for graph colouring (via backtracking).
- This speedup gets substantially smaller when considering hardware available today (e.g. $\sim 10^3$ for k-SAT).

Summary of results: good and bad news

- In the most optimistic hardware parameter regime, we could see speedup factors of $> 10^5$ (compared with a standard desktop PC) for k-SAT (via Grover's algorithm) and $> 10^4$ for graph colouring (via backtracking).
- This speedup gets substantially smaller when considering hardware available today (e.g. $\sim 10^3$ for k-SAT).
- If we additionally take into account the cost of classical error-correction processing, this speedup essentially disappears.

Summary of results: good and bad news

- In the most optimistic hardware parameter regime, we could see speedup factors of $> 10^5$ (compared with a standard desktop PC) for k-SAT (via Grover's algorithm) and $> 10^4$ for graph colouring (via backtracking).
- This speedup gets substantially smaller when considering hardware available today (e.g. $\sim 10^3$ for k-SAT).
- If we additionally take into account the cost of classical error-correction processing, this speedup essentially disappears.
- The number of physical qubits used is very large (e.g. $> 10^{12}$), almost all of which are used for fault-tolerance.

Summary of results: good and bad news

- In the most optimistic hardware parameter regime, we could see speedup factors of $> 10^5$ (compared with a standard desktop PC) for k-SAT (via Grover's algorithm) and $> 10^4$ for graph colouring (via backtracking).
- This speedup gets substantially smaller when considering hardware available today (e.g. $\sim 10^3$ for k-SAT).
- If we additionally take into account the cost of classical error-correction processing, this speedup essentially disappears.
- The number of physical qubits used is very large (e.g. $> 10^{12}$), almost all of which are used for fault-tolerance.
- This strongly motivates the design of improved fault-tolerance techniques!

Summary of results (1)

	Realistic	Plausible	Optimistic
Max n	65	72	78
T-depth	1.46×10^{12}	1.65×10^{13}	1.32×10^{14}
Toffoli count	4.41×10^{17}	5.52×10^{18}	4.79×10^{19}
Factory qubits	3.14×10^{13}	5.15×10^{12}	1.38×10^{12}
Speedup factor	1.62×10^{3}	1.73×10^{4}	1.83×10^{5}

Table : Likely speedup factors for 14-SAT via Grover's algorithm achievable in different regimes. Relative to an Intel Core i7-4790S CPU operating at 3.20GHz.

Summary of results (2)

	Realistic	Plausible	Optimistic
Max n	55	63	72
T-depth	1.63×10^{12}	1.43×10^{13}	1.63×10^{14}
T/Toffoli count	4.72×10^{18}	4.72×10^{19}	6.16×10^{20}
Factory qubits	3.85×10^{14}	5.03×10^{13}	2.17×10^{13}
Speedup factor	1.50×10^{1}	3.92×10^{2}	1.16×10^{4}

Table : Likely speedup factors for 12-SAT via backtracking achievable in different regimes.

Summary of results (3)

	Realistic	Plausible	Optimistic
Max n	113	128	144
T-depth	1.70×10^{12}	1.53×10^{13}	1.62×10^{14}
T/Toffoli count	8.51×10^{17}	1.02×10^{19}	1.28×10^{20}
Factory qubits	6.50×10^{13}	9.54×10^{12}	3.69×10^{12}
Speedup factor	7.25×10^{0}	5.17×10^{2}	4.16×10^{4}

Table : Likely speedup factors for graph colouring via backtracking achievable in different regimes.

Cost of classical processing

N	Realistic	Plausible	Optimistic
10^{12}	4.17×10^{7}	4.30×10^{4}	9.15×10^{-1}
10^{16}	2.29×10^{12}	7.76×10^{8}	2.23×10^{4}
10^{20}	3.10×10^{16}	3.07×10^{13}	3.28×10^{8}

Table : Classical processing required to implement N Toffoli gates under different regimes, based on extrapolation of runtimes reported by [Delfosse and Nickerson '17].

- Cost measured in processor-days (where type of processor is CPU, GPU and ASIC respectively in realistic, plausible and optimistic regimes).
- Assumes that the speedup offered by GPUs and ASICs over CPUs is a factor of 100 and 10^6 respectively.

Conclusions

We might be able to achieve quite a significant quantum speedup for common and practically relevant problems…

Conclusions

We might be able to achieve quite a significant quantum speedup for common and practically relevant problems…

…but there are some major challenges to be addressed before this becomes realistic. Improved fault-tolerance techniques would make a big difference.

Conclusions

We might be able to achieve quite a significant quantum speedup for common and practically relevant problems…

…but there are some major challenges to be addressed before this becomes realistic. Improved fault-tolerance techniques would make a big difference.

Thanks!

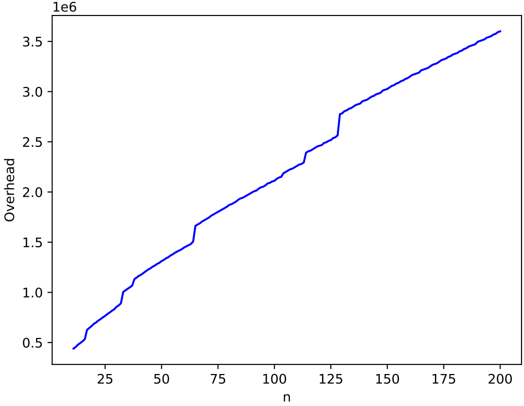


Figure : The runtime (circuit depth) of the quantum algorithm for backtracking is of the form $f(n,k)\sqrt{T}$, where T is the number of nodes in the backtracking tree. Figure illustrates scaling of $f(n,k)$ with n when k is chosen to be the expected chromatic number of a random graph.

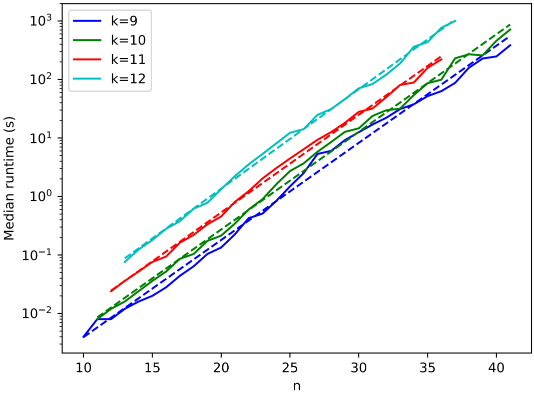


Figure : Runtime of the Maple_LCM_Dist SAT solver on random k-SAT instances with n variables and $\approx \alpha_k n$ clauses, where α_k is the satisfiability threshold. Solid line represents the median of at least 100 runs, in CPU-seconds. Dashed lines are linear least-squares fits.

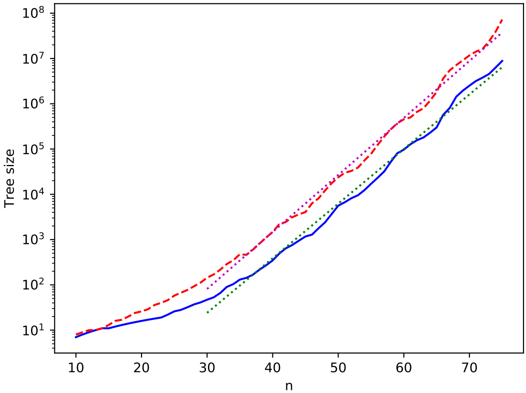


Figure : Number of nodes in the DSATUR k-colourability backtracking tree. Median (solid) and 90th percentile (dashed) over 1000 random graphs for each $n \in \{10, \ldots, 75\}$. Dotted lines are least-squares fits for the range $n \geqslant 30$.

Obtaining a quantum speedup

A prominent colouring algorithm, DSATUR, is based on the standard backtracking procedure, with optimisations:

- First find a large clique and colour it;
- At each step, select the most constrained vertex to colour.

Obtaining a quantum speedup

A prominent colouring algorithm, DSATUR, is based on the standard backtracking procedure, with optimisations:

- First find a large clique and colour it;
- At each step, select the most constrained vertex to colour.

We can accelerate DSATUR using a quantum algorithm:

Theorem (informal) [AM '15]

Let T be the number of nodes in the backtracking tree. Then there is a bounded-error quantum algorithm which runs in time $O(\sqrt{T}\,\mathrm{poly}(n))$ time and outputs whether or not an n-vertex graph is k-colourable.

How large a speedup is this in practice?

- For simplicity, we consider the case where we know an upper bound on T, and want to detect whether a k-colouring exists, rather than find one.

How large a speedup is this in practice?

- For simplicity, we consider the case where we know an upper bound on T, and want to detect whether a k-colouring exists, rather than find one.

- The algorithm applies phase estimation with precision $O(1/\sqrt{Tn})$ to a quantum walk in the backtracking tree.

How large a speedup is this in practice?

- For simplicity, we consider the case where we know an upper bound on T, and want to detect whether a k-colouring exists, rather than find one.
- The algorithm applies phase estimation with precision $O(1/\sqrt{Tn})$ to a quantum walk in the backtracking tree.
- The quantum walk alternates two operations, each corresponding to Grover-style diffusion D among nodes and their neighbours in the tree.

How large a speedup is this in practice?

- For simplicity, we consider the case where we know an upper bound on T, and want to detect whether a k-colouring exists, rather than find one.
- The algorithm applies phase estimation with precision $O(1/\sqrt{Tn})$ to a quantum walk in the backtracking tree.
- The quantum walk alternates two operations, each corresponding to Grover-style diffusion D among nodes and their neighbours in the tree.
- So the overall time taken by the algorithm is

$$C_P \times \sqrt{Tn} \times 2 \times T_D$$

where $C_P \leqslant 4$ is the constant from phase estimation.

Working out T_D

The time taken by this part of the algorithm is dominated by two classical operations:

1. Determine if a partial colouring is valid;
2. Find the most constrained uncoloured vertex in the graph.

Working out T_D

The time taken by this part of the algorithm is dominated by two classical operations:

1. Determine if a partial colouring is valid;
2. Find the most constrained uncoloured vertex in the graph.

- We can find low-depth quantum circuits for each of these using, e.g., efficient quantum circuits for integer arithmetic [Draper et al '04].

Working out T_D

The time taken by this part of the algorithm is dominated by two classical operations:

1. Determine if a partial colouring is valid;
2. Find the most constrained uncoloured vertex in the graph.

- We can find low-depth quantum circuits for each of these using, e.g., efficient quantum circuits for integer arithmetic [Draper et al '04].

- Guiding principle: Minimise the T-depth ($\Rightarrow$ runtime) of the resulting quantum circuits.

Working out T_D

The time taken by this part of the algorithm is dominated by two classical operations:

1. Determine if a partial colouring is valid;
2. Find the most constrained uncoloured vertex in the graph.

- We can find low-depth quantum circuits for each of these using, e.g., efficient quantum circuits for integer arithmetic [Draper et al '04].

- Guiding principle: Minimise the T-depth ($\Rightarrow$ runtime) of the resulting quantum circuits.

We can achieve $T_D \leqslant 3200$ for colouring a $\leqslant$136-vertex graph.