Combinatorial optimization on quantum computers

Ashley Montanaro

Phasecraft Ltd
School of Mathematics, University of Bristol

University of BRISTOL
EPSRC Engineering and Physical Sciences Research Council
erc European Research Council Established by the European Commission
PHASECRAFT

Combinatorial optimization

Combinatorial optimization problems are characterised by needing to search over exponentially many possible solutions.

For example:

- Colouring a graph with the minimal number of colours such that no adjacent vertices share a colour;
- Finding the lowest-cost route that visits all of a set of cities;
- Determining if a system of linear equations over integers $\{0, 1\}$ has a solution.

Quantum computers can sometimes achieve a speedup in solving optimization problems over our best classical algorithms.

However, the speedup (when provable) is usually at most quadratic, as opposed to exponential (e.g. running time $2^n \to 2^{n/2}$)

Today's talk

Today I will discuss some quantum algorithms for solving optimization and constraint satisfaction problems (CSPs):

- Quantum speedup of backtracking and branch-and-bound algorithms [AM '15, AM '19] (with a large-scale, fault-tolerant quantum computer)

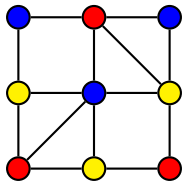
- Quantum speedup of boolean satisfability [Boulebnane and AM '23] (possibly with a near-term quantum computer)

Solving hard constraint satisfaction problems with structure

Often we can achieve a better complexity than unstructured search or optimization by using the structure of the problem we need to solve.

One of the most prominent techniques for this is backtracking ("trial and error").

We can illustrate backtracking with graph k-colouring:

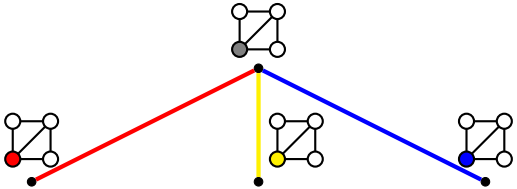


An NP-complete problem with a huge number of direct applications, including register allocation; scheduling; frequency assignment problems; ...

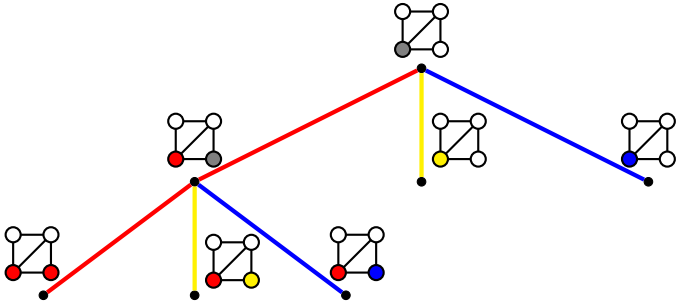
Colouring by backtracking

Colouring by backtracking

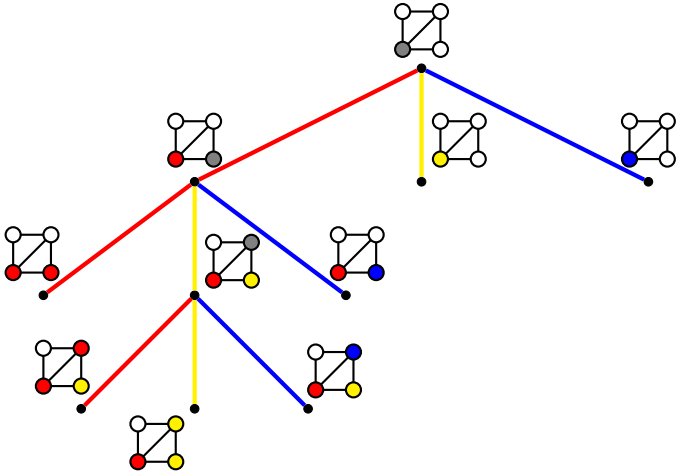
Colouring by backtracking



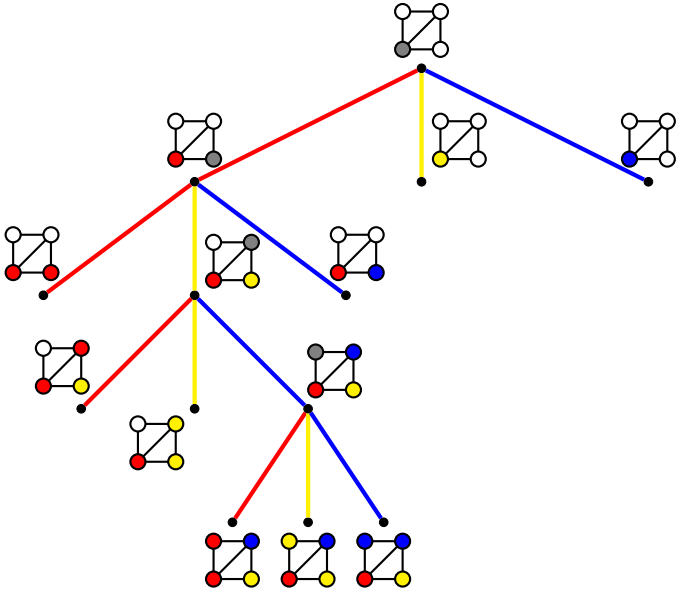
Colouring by backtracking



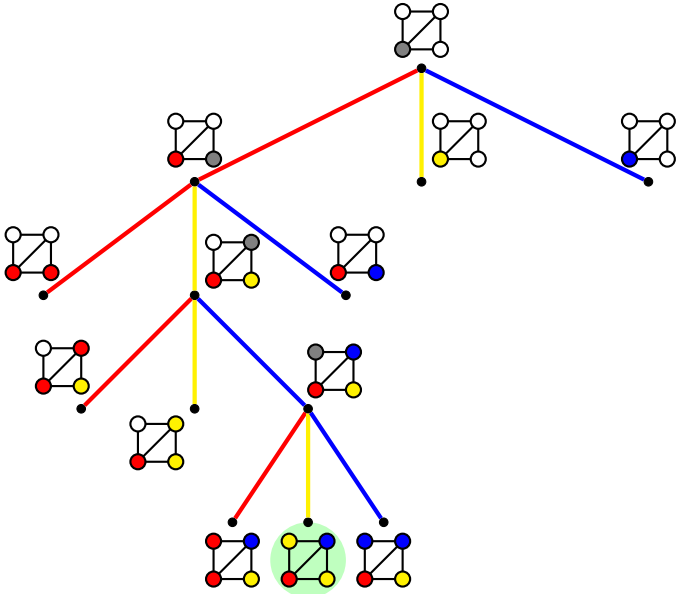
Colouring by backtracking



Colouring by backtracking

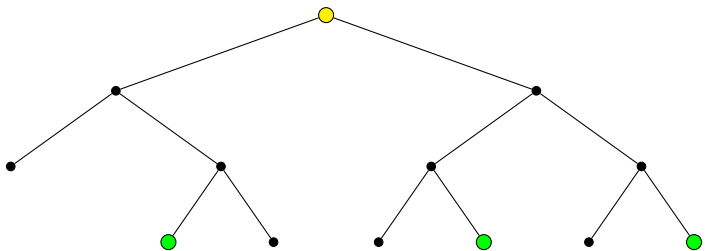


Colouring by backtracking



Search in a tree

Imagine we want to find a "marked" vertex in a tree where we only have local knowledge, starting from the root.

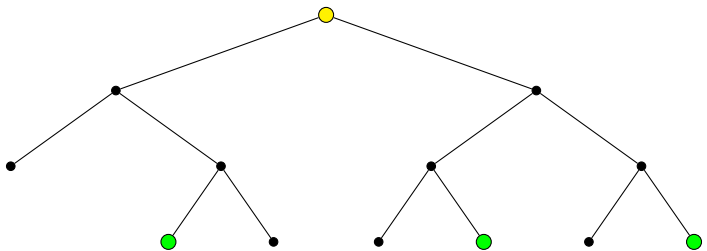


If the tree has T vertices, this requires $\sim T$ time classically in the worst case.

Quantum search in a tree

Theorem [Belovs '13]

There is a quantum algorithm that can detect existence of a marked vertex in a tree with T vertices and depth d, using $O(\sqrt{Td})$ queries.



The algorithm is based on a quantum walk in the tree.

From quantum search in trees to backtracking

- A backtracking algorithm solving a problem with n variables explores a tree of size T and depth n.
- The quantum walk algorithm for search in trees can be applied, yielding:

Theorem (informal) [AM '18]

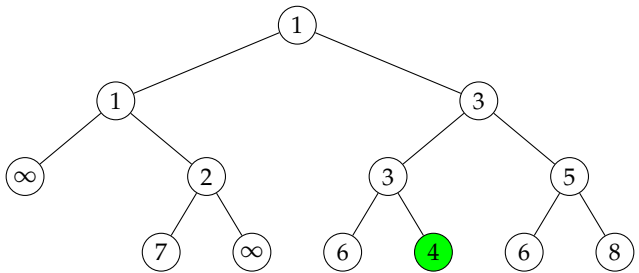
There is a corresponding quantum algorithm which finds a solution, or outputs that one does not exist, in time $O(\sqrt{T}\,\mathrm{poly}(n))$, with 1% probability of error.

- We normally think of T as exponential in n; in this case, the speedup is near-quadratic.
- Subsequent improvements to this algorithm: [Ambainis and Kokainis '17], [Jarret and Wan '18]

Branch-and-bound algorithms

The backtracking approach can be generalised to solve optimization problems via a technique known as branch-and-bound, which is applicable whenever we have:

- A branching procedure that splits a set of potential solutions into subsets;
- A bounding procedure that returns a lower bound on the cost of any solution in a subset.



Quantum speedup of branch-and-bound

Theorem (informal) [AM'19]

Assume there is a classical algorithm that solves an optimization problem using the branch and bound procedures T times. Then there is a quantum algorithm that solves the same problem using these procedures $O(\sqrt{T})$ times (up to lower-order terms).

- The quantum algorithm is based on the use of the backtracking algorithm as a subroutine.
- It can be applied to find ground states of the Ising model (aka Max-Cut):

$$\min_{z\in\{\pm 1\}^n} \sum_{i<j} a_{ij} z_i z_j$$

- e.g. Sherrington-Kirkpatrick model $a_{ij} \sim N(0,1)$: runtime $O(2^{0.226n})$ or better, beating Grover search $O(2^{0.5n})$.

Other developments in quantum backtracking

Applications:

- lattice-based cryptography, e.g. [Alkim et al '16, del Pino et al '16]
- Travelling Salesman Problem [Moylett et al '17]
- exact satisfiability [Mandrà et al '16]
- constraint programming [Booth et al '21]

Experimental implementation (in simulation) for 2-colouring a graph with 4 vertices [Martiel and Remaud '19]

Do these theoretical speedups translate into real-world speedups?

Imagine we want to solve a problem within a runtime of at most 1 day.

- In an optimistic hardware parameter regime, we could see speedup factors of $> 10^4$ (compared with a standard desktop PC) [Campbell et al '19]
- This speedup gets substantially smaller when considering parameters corresponding to quantum hardware available today.
- If we additionally take into account the cost of classical error-correction processing, this speedup 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!

Boolean satisfiability

Are there hard constraint satisfaction problems which we may be able to solve using quantum computers in the near term?

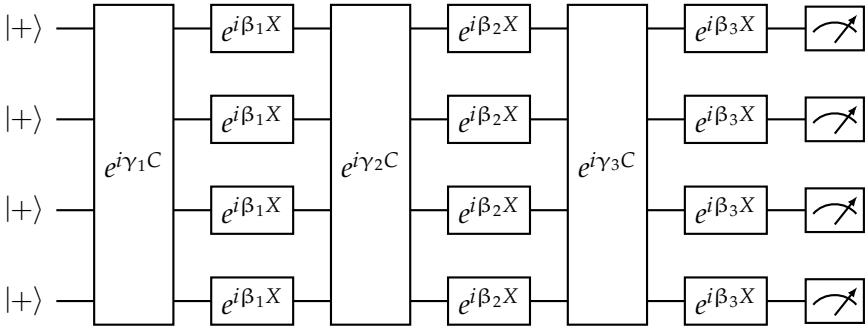
Consider the boolean satisfiability problem: finding a solution to a set of constraints on boolean variables $x_1, \ldots, x_n$ of the form

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_4 \vee x_7) \wedge (x_2 \vee \neg x_5 \vee \neg x_7)$$

Here: the special case of the random k-SAT problem where each clause contains k variables and is chosen at random.

The quantum approximate optimization algorithm (QAOA)

We will apply a quantum algorithm known as QAOA [Hogg '00, Farhi et al '14] for minimising a cost function $C : \{0,1\}^n \to \mathbb{Z}$.



The quantum approximate optimization algorithm (QAOA)

Mathematically, QAOA looks like this:

- Alternate between evolution under (diagonal) cost Hamiltonian and transverse field Hamiltonian.

$$|\Psi_{\mathrm{QAOA}}(\boldsymbol{\beta},\boldsymbol{\gamma})\rangle := e^{-i\beta_p H_B} e^{-i\gamma_p H_C} \dots e^{-i\beta_1 H_B} e^{-i\gamma_1 H_C} |+\rangle^{\otimes n},$$

$$H_C := \sum_{z\in\{0,1\}^n} C(z)\,|z\rangle\langle z|,$$

$$H_B := \sum_{1\leqslant j\leqslant n} X_j,$$

  where C is an integer-valued numerical function.

- After evolution, measure state in computational basis to obtain candidate solution.
- Variational quantum algorithm: parameters $\boldsymbol{\beta}, \boldsymbol{\gamma}$ ("QAOA angles") need to be determined.

Applying QAOA to k-SAT

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_4 \vee x_7) \wedge (x_2 \vee \neg x_5 \vee \neg x_7)$$

- Cost function $C(z)$ = number of violated clauses.

$$\begin{aligned}
\text{Problem instance:} \quad & \boldsymbol{\sigma} = (\sigma_0, \ldots, \sigma_{m-1}) \\
\text{Cost function:} \quad & C_{\boldsymbol{\sigma}}(z) := \sum_{j \in [m]} \mathbf{1}\left[z \text{ violates } \sigma_j\right] \\
\text{Diagonal Hamiltonian:} \quad & H[\boldsymbol{\sigma}] := \sum_{z \in \{0,1\}^n} C_{\boldsymbol{\sigma}}(z)\, |z\rangle \langle z|
\end{aligned}$$

- We wish to maximize the success probability $p_{\text{success}}(\boldsymbol{\sigma}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ (not the expected number of satisfied constraints!):

$$\max_{\boldsymbol{\beta}, \boldsymbol{\gamma}} \langle \Psi_{\text{QAOA}}(\boldsymbol{\sigma}, \boldsymbol{\beta}, \boldsymbol{\gamma}) | \Pi_{\ker H[\boldsymbol{\sigma}]} | \Psi_{\text{QAOA}}(\boldsymbol{\sigma}, \boldsymbol{\beta}, \boldsymbol{\gamma}) \rangle$$

- We do this by optimizing the parameters over random small instances.

Analytic estimate for the success probability

For random instances of k-SAT with an expected number of clauses $m = rn$, we show:

Exponential scaling of expected success probability

Let $k = 2^q \geqslant 2$ be an integer, $p \geqslant 1$ an integer and $\boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathbf{R}^p$.
For $\boldsymbol{\gamma}$ sufficiently small (independent of instance size n),

$$\lim_{n\to\infty} \frac{1}{n} \log \mathbf{E}_{\boldsymbol{\sigma}\sim \mathsf{kSAT}(n,r,2^q)} \left[p_{\mathsf{success}}(\boldsymbol{\sigma}, \boldsymbol{\beta}, \boldsymbol{\gamma})\right]$$

exists and can be computed from the fixed point of a function of 2^{2p+1} complex variables.

The scaling exponent of the expected success probability calculated by the procedure can be used as a proxy to optimize the success probability over $\boldsymbol{\beta}, \boldsymbol{\gamma}$.

Analytic vs. empirical scaling exponents

How well do analytic predictions match empirical results?

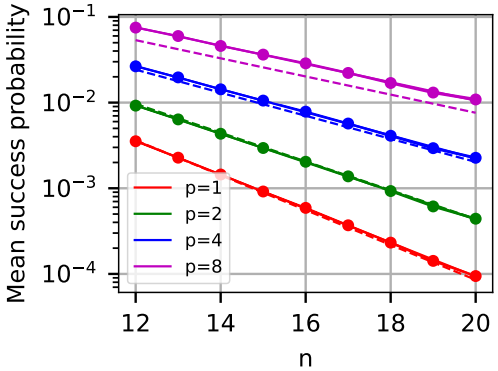
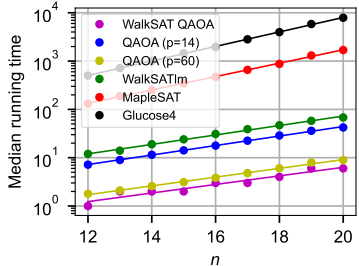


Figure: Analytic and empirical exponents for fixed $k = 8$ and varying p. Empirical exponents were fitted from empirical average success probabilities (solid lines). Analytic fits (dashed lines) were represented assuming a success probability 1 "at $n = 0$".

QAOA vs. classical algorithms

- We benchmarked QAOA against many classical SAT solvers; WalkSATlm found to perform best.

- Method: similar to benchmarks above but consider median running time, not average-instance success probability.



Solver	Fit
WalkSAT QAOA	$-3.232 + 0.295n$
QAOA ($p = 14$)	$-1.064 + 0.326n$
QAOA ($p = 60$)	$-2.842 + 0.302n$
`walksatlm`	$-0.309 + 0.325n$
`maplesat`	$1.531 + 0.461n$
`glucose4`	$2.998 + 0.498n$

Figure: Scaling behaviour of median running times of selected classical and quantum algorithms for 8-SAT. WalkSAT QAOA uses $p = 60$.

QAOA vs. WalkSATlm

How deep a quantum algorithm will we need to outperform classical algorithms?

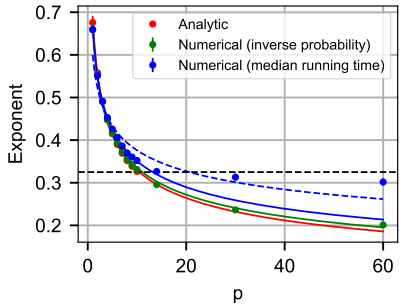


Figure: Running times of QAOA compared with WalkSATlm for random 8-SAT. Blue dashed line is fitting based on all p, blue solid line is using $p \leqslant 10$.

Conclusions

We might be able to achieve a fairly significant quantum speedup for common and practically relevant combinatorial optimization problems...
...but we're not quite there yet! More work remains to be done on algorithm design and applications.

Advert: We are hiring at Phasecraft for quantum algorithms scientists to work on quantum optimization algorithms.

Further reading:

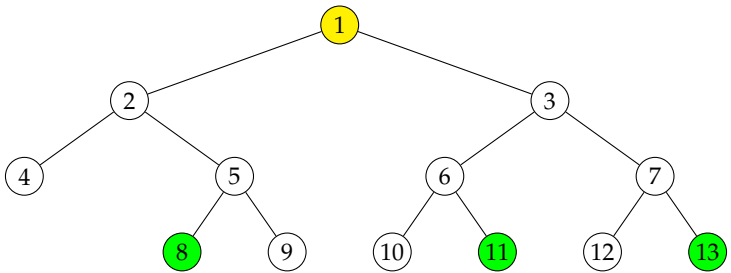
- Backtracking: arXiv:1509.02374, arXiv:1906.10375
- k-SAT: arXiv:2208.06909 (with Sami Boulebnane)
- Survey on quantum optimization algorithms: arXiv:2312.02279 (Abbas et al)

Thanks!

Quantum walk in a tree

For each vertex x with c children, define

$$|\psi_x\rangle = \frac{1}{\sqrt{c+1}}\left(|x\rangle + \sum_{\text{children } y} |y\rangle\right), \quad D_x = \begin{cases} I - 2\,|\psi_x\rangle\langle\psi_x| & [x \text{ unmarked}] \\ I & [x \text{ marked}]. \end{cases}$$

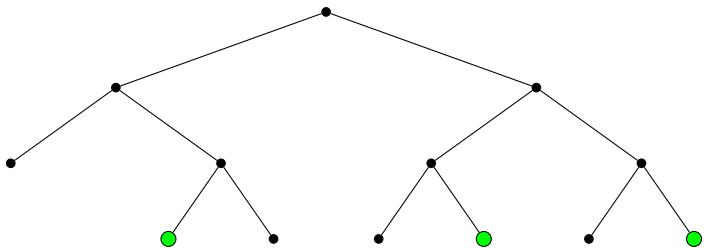


e.g. $\quad |\psi_2\rangle = \frac{1}{\sqrt{3}}\left(|2\rangle + |4\rangle + |5\rangle\right), \quad D_2 = \frac{1}{3}\begin{pmatrix} 1 & -2 & -2 \\ -2 & 1 & -2 \\ -2 & -2 & 1 \end{pmatrix}$

Quantum walk in a tree

Let A and B be the sets of vertices an even and odd distance from the root, respectively.

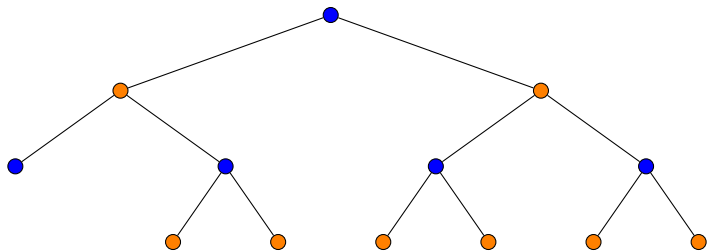
Then a step of the walk is applying the unitary operator $W = D_B D_A$, where $D_A = \bigoplus_{x \in A} D_x$ and $D_B = \bigoplus_{x \in B} D_x$.



Quantum walk in a tree

Let A and B be the sets of vertices an even and odd distance from the root, respectively.

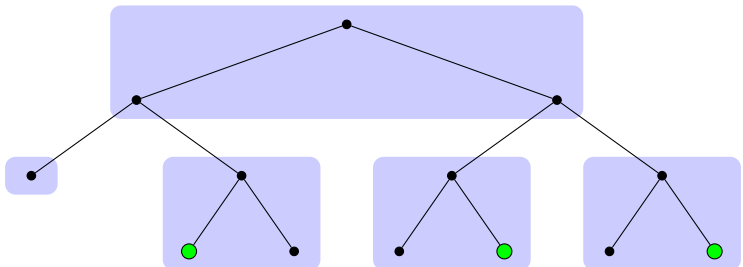
Then a step of the walk is applying the unitary operator $W = D_B D_A$, where $D_A = \bigoplus_{x \in A} D_x$ and $D_B = \bigoplus_{x \in B} D_x$.



Quantum walk in a tree

Let A and B be the sets of vertices an even and odd distance from the root, respectively.

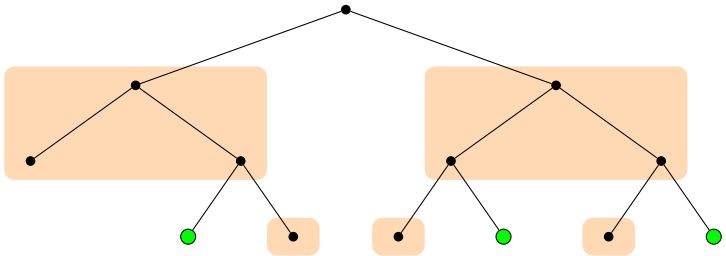
Then a step of the walk is applying the unitary operator $W = D_B D_A$, where $D_A = \bigoplus_{x \in A} D_x$ and $D_B = \bigoplus_{x \in B} D_x$.



Quantum walk in a tree

Let A and B be the sets of vertices an even and odd distance from the root, respectively.

Then a step of the walk is applying the unitary operator $W = D_B D_A$, where $D_A = \bigoplus_{x \in A} D_x$ and $D_B = \bigoplus_{x \in B} D_x$.



Analysing the quantum walk

Claims

1. $\geqslant 1$ marked vertex: W has an eigenvector w/eigenvalue 1.
2. No marked vertex: W has no eigenvector w/eigenvalue 1.

Quantum computers can estimate eigenvalues via a procedure called phase estimation, allowing these two cases to be distinguished.

Small print: We actually need to apply this to a similar unitary operator W', such that the root $|r\rangle$ is close to an eigenvector of W'.

W' has an approximate "phase gap" of $\sim 1/\sqrt{Td}$, allowing the two cases to be distinguished with $O(\sqrt{Td})$ uses of W'.

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.

Summary of results

	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	8.24×10^{17}	9.94×10^{18}	1.24×10^{20}
Factory qubits	6.29×10^{13}	9.26×10^{12}	3.59×10^{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.

Complexity estimates for other algorithms and CSPs were obtained by [Sanders et al '20].