The Surface Code and its Decoders

Ben Criger [1]

[1] Qutech, TU Delft

November 11, 2016

Introduction

Why is (Quantum) Error Correction Necessary?

Classical computers are based on continuous physical states but discrete logical states:

Why is (Quantum) Error Correction Necessary?

Classical computers are based on continuous physical states but discrete logical states:

Why is (Quantum) Error Correction Necessary?

Classical computers are based on continuous physical states but discrete logical states:

Small deviations in these parameters can be measured and corrected accurately.

Why is (Quantum) Error Correction Necessary?

Classical computers are based on continuous physical states but discrete logical states:

Small deviations in these parameters can be measured and corrected accurately.

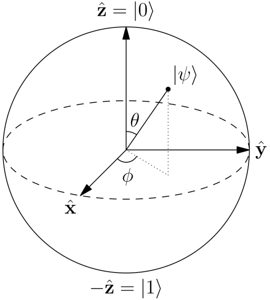
Quantum computing, however:

Why is (Quantum) Error Correction Necessary?

Classical computers are based on continuous physical states but discrete logical states:

Small deviations in these parameters can be measured and corrected accurately.

Quantum computing, however:



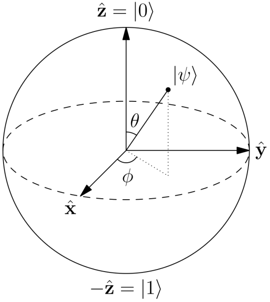
- has states which are inherently continuous

Why is (Quantum) Error Correction Necessary?

Classical computers are based on continuous physical states but discrete logical states:

Small deviations in these parameters can be measured and corrected accurately.

Quantum computing, however:



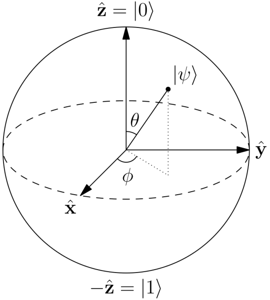
- has states which are inherently continuous
- cannot use direct measurement to sense deviations

Why is (Quantum) Error Correction Necessary?

Classical computers are based on continuous physical states but discrete logical states:

Small deviations in these parameters can be measured and corrected accurately.

Quantum computing, however:



- has states which are inherently continuous
- cannot use direct measurement to sense deviations
- relies on operations that are continuous with finite accuracy

What are (Quantum) Error Correcting Codes?

A code is a subset of possible symbols we could send through a noisy channel.

What are (Quantum) Error Correcting Codes?

A code is a subset of possible symbols we could send through a noisy channel.

Linear codes are subspaces of a vector space, most quantum codes are subspaces of $\mathcal{H} = \mathbb{C}_2^n$.

What are (Quantum) Error Correcting Codes?

A code is a subset of possible symbols we could send through a noisy channel.

Linear codes are subspaces of a vector space, most quantum codes are subspaces of $\mathcal{H} = \mathbb{C}_2^n$.

Stabiliser codes are subspaces specified by lists of Pauli matrices, they are the mutual $+1$ eigenspaces of these matrices:

What are (Quantum) Error Correcting Codes?

A code is a subset of possible symbols we could send through a noisy channel.

Linear codes are subspaces of a vector space, most quantum codes are subspaces of $\mathcal{H} = \mathbb{C}_2^n$.

Stabiliser codes are subspaces specified by lists of Pauli matrices, they are the mutual $+1$ eigenspaces of these matrices:

$$S = \langle ZZI, \, XXX \rangle \quad |\psi\rangle = \alpha \left(\frac{|001\rangle + |110\rangle}{\sqrt{2}} \right) + \beta \left(\frac{|000\rangle + |111\rangle}{\sqrt{2}} \right)$$

What are (Quantum) Error Correcting Codes?

A code is a subset of possible symbols we could send through a noisy channel.

Linear codes are subspaces of a vector space, most quantum codes are subspaces of $\mathcal{H} = \mathbb{C}_2^n$.

Stabiliser codes are subspaces specified by lists of Pauli matrices, they are the mutual $+1$ eigenspaces of these matrices:

$$S = \langle ZZI, \, XXX \rangle \quad |\psi\rangle = \alpha \left(\frac{|001\rangle + |110\rangle}{\sqrt{2}} \right) + \beta \left(\frac{|000\rangle + |111\rangle}{\sqrt{2}} \right)$$

We measure the stabilisers onto ancilla systems, and obtain a *syndrome* (a list of things wrong with the state) that does not depend on the code state, only on the errors that have occurred.

What are (Quantum) Error Correcting Codes?

A code is a subset of possible symbols we could send through a noisy channel.

Linear codes are subspaces of a vector space, most quantum codes are subspaces of $\mathcal{H} = \mathbb{C}_2^n$.

Stabiliser codes are subspaces specified by lists of Pauli matrices, they are the mutual $+1$ eigenspaces of these matrices:

$$S = \langle ZZI, \, XXX \rangle \quad |\psi\rangle = \alpha \left(\frac{|001\rangle + |110\rangle}{\sqrt{2}} \right) + \beta \left(\frac{|000\rangle + |111\rangle}{\sqrt{2}} \right)$$

We measure the stabilisers onto ancilla systems, and obtain a *syndrome* (a list of things wrong with the state) that does not depend on the code state, only on the errors that have occurred.

The process of inferring which error happened given a syndrome is called *decoding*.

What are Fault-Tolerant Operations?

Fault-tolerant operations can be carried out on encoded data, using multiple redundant gates.

What are Fault-Tolerant Operations?

Fault-tolerant operations can be carried out on encoded data, using multiple redundant gates.
If any one gate fails (does something random), we can use error-correction to recover.

What are Fault-Tolerant Operations?

Fault-tolerant operations can be carried out on encoded data, using multiple redundant gates.

If any one gate fails (does something random), we can use error-correction to recover.

These faults can cause the syndrome to be incorrect.

What are Fault-Tolerant Operations?

Fault-tolerant operations can be carried out on encoded data, using multiple redundant gates.

If any one gate fails (does something random), we can use error-correction to recover.

These faults can cause the syndrome to be incorrect.

If we perform stabiliser measurement repeatedly, we can detect syndrome errors at times where the syndrome measurement is different than the previous round. This data can be used like a syndrome to correct the measurement errors.

What are Fault-Tolerant Operations?

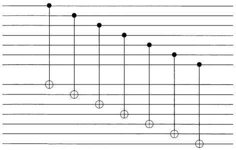
Fault-tolerant operations can be carried out on encoded data, using multiple redundant gates.

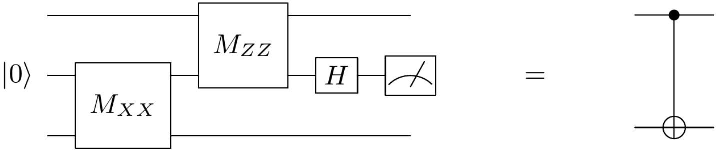
If any one gate fails (does something random), we can use error-correction to recover.

These faults can cause the syndrome to be incorrect.

If we perform stabiliser measurement repeatedly, we can detect syndrome errors at times where the syndrome measurement is different than the previous round. This data can be used like a syndrome to correct the measurement errors.

Fault-tolerant gates are typically either *transversal* or *measurement-based*:





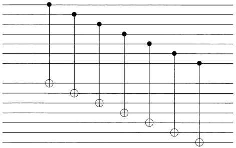
What are Fault-Tolerant Operations?

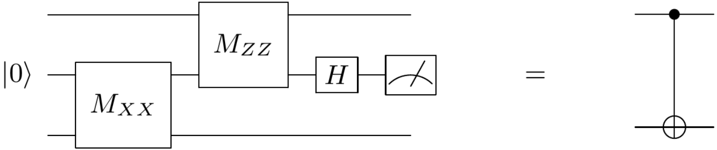
Fault-tolerant operations can be carried out on encoded data, using multiple redundant gates.

If any one gate fails (does something random), we can use error-correction to recover. These faults can cause the syndrome to be incorrect.

If we perform stabiliser measurement repeatedly, we can detect syndrome errors at times where the syndrome measurement is different than the previous round. This data can be used like a syndrome to correct the measurement errors.

Fault-tolerant gates are typically either *transversal* or *measurement-based*:



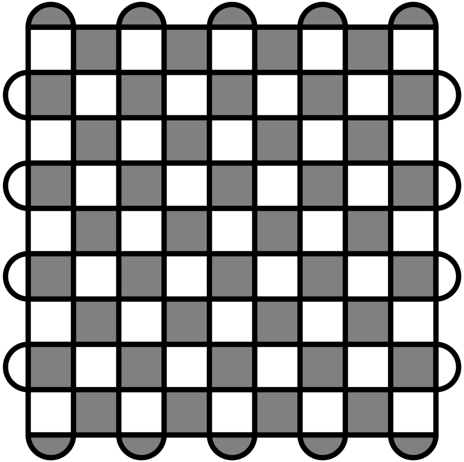


We'll focus on fault-tolerant quantum memory for now.

The Surface Code

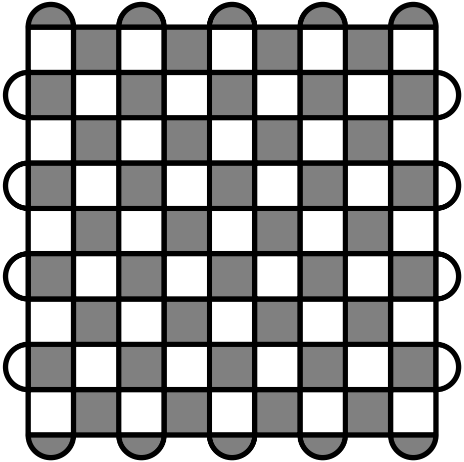
Requirements

The stabilisers of the surface code are local, constant-weight, and can be measured with a planar circuit:



Requirements

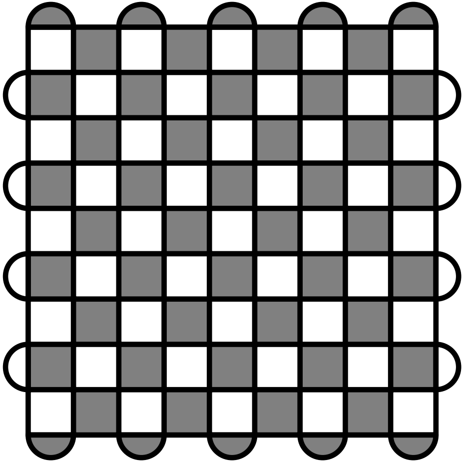
The stabilisers of the surface code are local, constant-weight, and can be measured with a planar circuit:



- $[\![n, k, d]\!] = [\![l^2, 1, l]\!]$ ("half" of a toric code)

Requirements

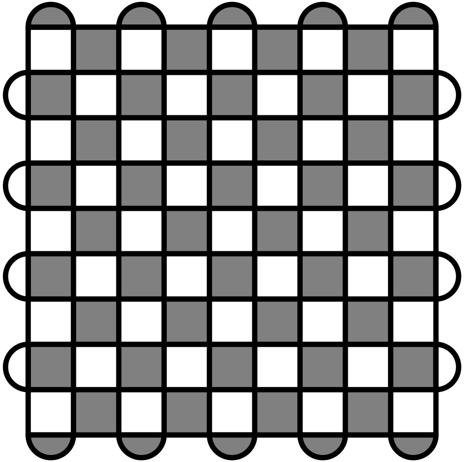
The stabilisers of the surface code are local, constant-weight, and can be measured with a planar circuit:



- $[\![n, k, d]\!] = [\![l^2, 1, l]\!]$ (“half” of a toric code)
- Qubits associated with vertices

Requirements

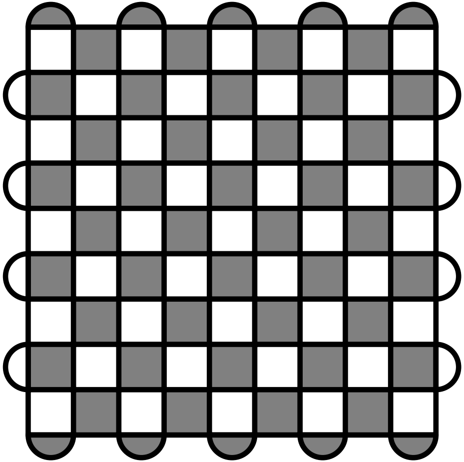
The stabilisers of the surface code are local, constant-weight, and can be measured with a planar circuit:



- $[\![n,\, k,\, d]\!] = [\![l^2,\, 1,\, l]\!]$ ("half" of a toric code)
- Qubits associated with vertices
- $\boxed{}$: weight-4 X check

Requirements

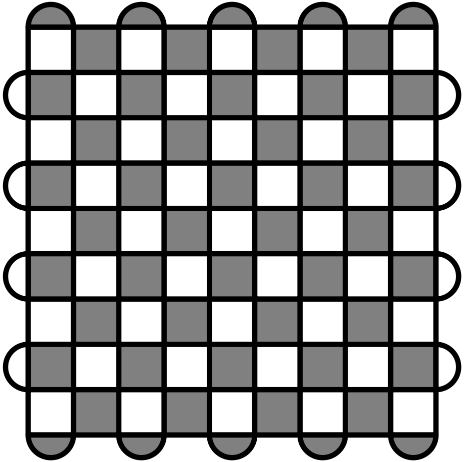
The stabilisers of the surface code are local, constant-weight, and can be measured with a planar circuit:



- $[\![n, k, d]\!] = [\![l^2, 1, l]\!]$ ("half" of a toric code)
- Qubits associated with vertices
- $\square$: weight-4 X check
- $\blacksquare$: weight-4 Z check

Requirements

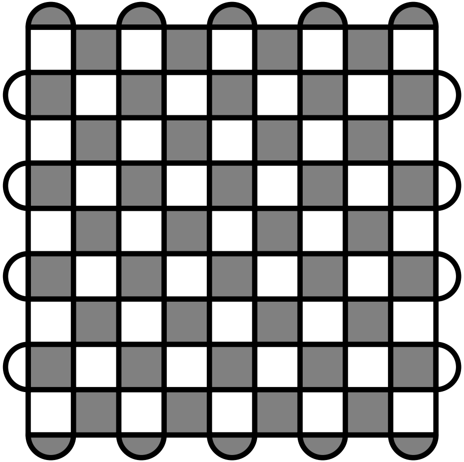
The stabilisers of the surface code are local, constant-weight, and can be measured with a planar circuit:



- $[\![n,\, k,\, d]\!] = [\![l^2,\, 1,\, l]\!]$ (“half” of a toric code)
- Qubits associated with vertices
-  : weight-4 X check
-  : weight-4 Z check
- Weight-2 boundary checks: , 

Requirements

The stabilisers of the surface code are local, constant-weight, and can be measured with a planar circuit:



- $[\![n,\,k,\,d]\!] = [\![l^2,\,1,\,l]\!]$ ("half" of a toric code)
- Qubits associated with vertices
-  : weight-4 X check
-  : weight-4 Z check
- Weight-2 boundary checks: , 
- On average, can protect against $\sim 1\%$ of operations failing, as long as the failure mode is purely decoherent (not over/under-rotation).

Pros and Cons

Pros:

Pros and Cons

Pros:

- It's well understood how to compute with the surface code, exotic error models and large surface-based computations have been considered.

Pros and Cons

Pros:

- It’s well understood how to compute with the surface code, exotic error models and large surface-based computations have been considered.
- The high threshold and ease of use make it favourable for near-term, on-chip implementations.

Pros and Cons

Pros:

- It's well understood how to compute with the surface code, exotic error models and large surface-based computations have been considered.
- The high threshold and ease of use make it favourable for near-term, on-chip implementations.

Cons:

Pros and Cons

Pros:

- It's well understood how to compute with the surface code, exotic error models and large surface-based computations have been considered.
- The high threshold and ease of use make it favourable for near-term, on-chip implementations.

Cons:

- No transversal logical operations other than Paulis. Cliffords take time $\mathcal{O}(d)$ to perform by repeated measurement, non-Clifford operations have higher overhead.

Pros and Cons

Pros:

- It's well understood how to compute with the surface code, exotic error models and large surface-based computations have been considered.
- The high threshold and ease of use make it favourable for near-term, on-chip implementations.

Cons:

- No transversal logical operations other than Paulis. Cliffords take time $\mathcal{O}(d)$ to perform by repeated measurement, non-Clifford operations have higher overhead.
- The surface code doesn't compare well with the bounds on how well a quantum code can function. $^k/_n$ can approach a constant as $n \to \infty$, but for the surface code, it decays to 0. This implies that large computations have large overhead.

Pros and Cons

Pros:

- It's well understood how to compute with the surface code, exotic error models and large surface-based computations have been considered.
- The high threshold and ease of use make it favourable for near-term, on-chip implementations.

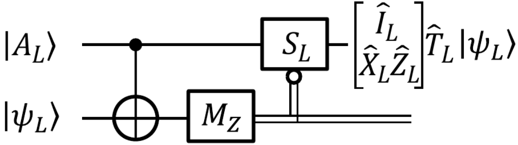
Cons:

- No transversal logical operations other than Paulis. Cliffords take time $\mathcal{O}(d)$ to perform by repeated measurement, non-Clifford operations have higher overhead.
- The surface code doesn't compare well with the bounds on how well a quantum code can function. k/n can approach a constant as $n \to \infty$, but for the surface code, it decays to 0. This implies that large computations have large overhead.
- Most urgent disadvantage is that frequently-used decoding algorithms take an amount of time that scales (provably) as d^{6-12}, resulting in a backlog.

The Backlog Problem

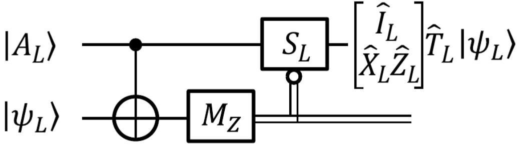
The Backlog Problem

To produce a universal gate set, some gates are performed by *gate teleportation*:



The Backlog Problem

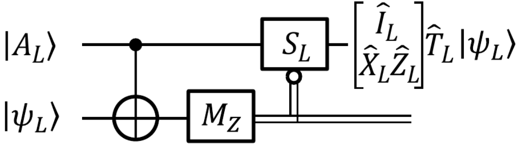
To produce a universal gate set, some gates are performed by *gate teleportation*:



This requires us to feed a logical measurement eigenvalue forward, which requires us to decode.

The Backlog Problem

To produce a universal gate set, some gates are performed by *gate teleportation*:

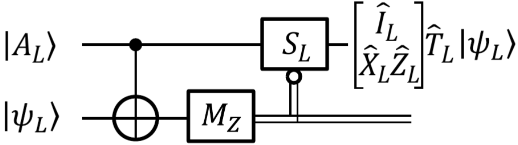


This requires us to feed a logical measurement eigenvalue forward, which requires us to decode.

The rest of the logical qubits have to be measured during this time, producing $\mathcal{O}(d^6)$ syndrome rounds.

The Backlog Problem

To produce a universal gate set, some gates are performed by *gate teleportation*:



This requires us to feed a logical measurement eigenvalue forward, which requires us to decode.

The rest of the logical qubits have to be measured during this time, producing $\mathcal{O}(d^6)$ syndrome rounds.

Repeated feed-forward results in computation time scaling like a power tower.
Exercise: What happens if the decoder takes time $O(d)$?

Fast Decoders for Topological Quantum Codes

Known Decoders

Renormalisation Group

Fast Decoders for Topological Quantum Codes

Guillaume Duclos-Cianci and David Poulin
Département de Physique, Université de Sherbrooke, Québec, Canada
(Dated: February 5, 2010)

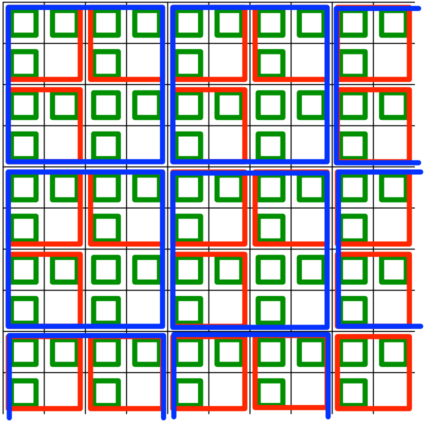
We present a family of algorithms, combining real-space renormalization methods and belief propagation, to estimate the free energy of a topologically ordered system in the presence of defects. Such an algorithm is needed to preserve the quantum information stored in the ground space of a topologically ordered system and to decode topological error-correcting codes. For a system of linear size ℓ, our algorithm runs in time $\log \ell$ compared to ℓ^6 needed for the minimum-weight perfect matching algorithm previously used in this context and achieves a higher depolarizing error threshold.

Inherently parallel algorithms exist to reduce the decoding time to $\log(d)$.

Known Decoders

Renormalisation Group

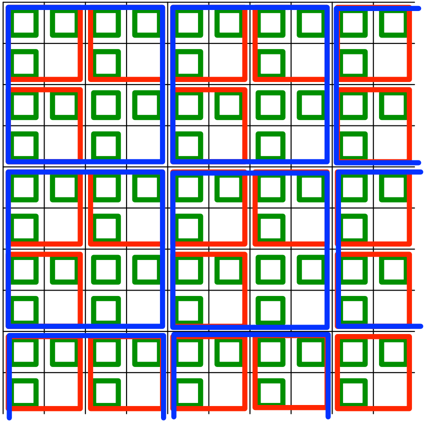
This speedup requires us to use a local lookup table and modified belief propagation to achieve a parallelizable algorithm:



Known Decoders

Renormalisation Group

This speedup requires us to use a local lookup table and modified belief propagation to achieve a parallelizable algorithm:

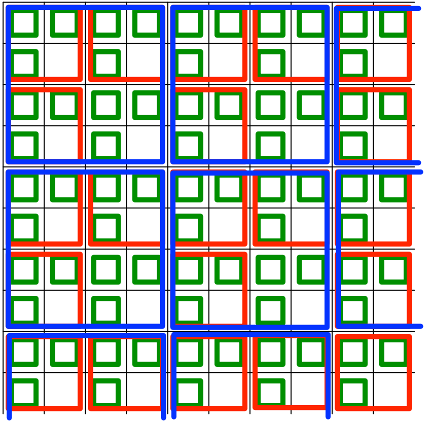


- data-only thresholds: $\sim 8\%$ (optimal: $\sim 10\%$)

Known Decoders

Renormalisation Group

This speedup requires us to use a local lookup table and modified belief propagation to achieve a parallelizable algorithm:

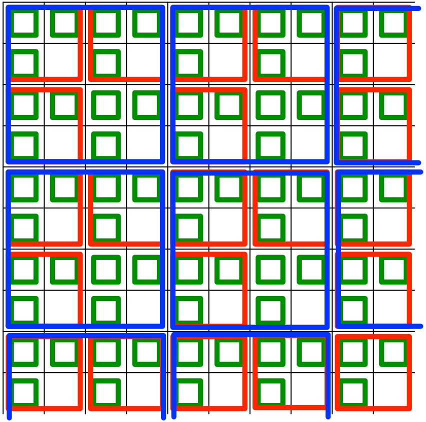


- data-only thresholds: $\sim 8\%$ (optimal: $\sim 10\%$)
- data & syndrome thresholds: $\sim 1.9\%$ (optimal: $\sim 3\%$)

Known Decoders

Renormalisation Group

This speedup requires us to use a local lookup table and modified belief propagation to achieve a parallelizable algorithm:

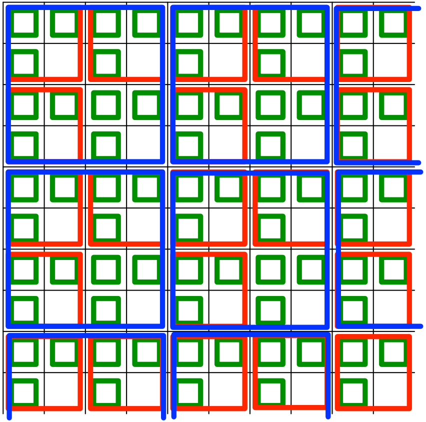


- data-only thresholds: $\sim 8\%$ (optimal: $\sim 10\%$)
- data & syndrome thresholds: $\sim 1.9\%$ (optimal: $\sim 3\%$)
- circuit-based thresholds: unknown

Known Decoders

Renormalisation Group

This speedup requires us to use a local lookup table and modified belief propagation to achieve a parallelizable algorithm:

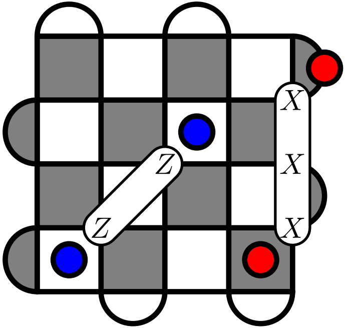


- data-only thresholds: $\sim 8\%$ (optimal: $\sim 10\%$)
- data & syndrome thresholds: $\sim 1.9\%$ (optimal: $\sim 3\%$)
- circuit-based thresholds: unknown
- effect of accurate error modelling: unknown

Known Decoders

'Interleaved' MWPM

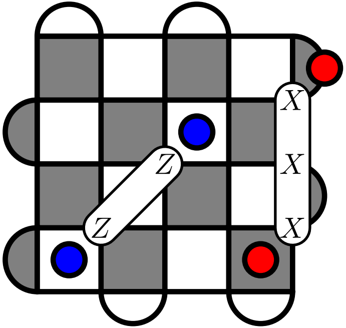
The syndromes of the surface code appear at the ends of chains of errors:



Known Decoders

'Interleaved' MWPM

The syndromes of the surface code appear at the ends of chains of errors:

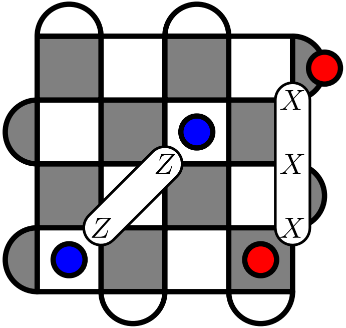


- To determine which error occurred, we have to 'match' the syndromes into pairs.

Known Decoders

'Interleaved' MWPM

The syndromes of the surface code appear at the ends of chains of errors:

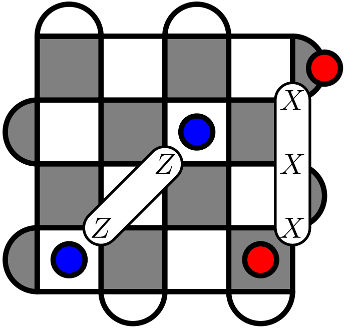


- To determine which error occurred, we have to 'match' the syndromes into pairs.
- We base this decision on the probability of a chain connecting two syndromes, which we get from the error model.

Known Decoders

'Interleaved' MWPM

The syndromes of the surface code appear at the ends of chains of errors:



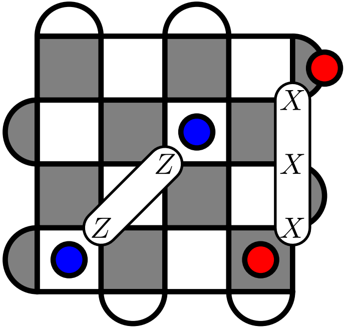
- To determine which error occurred, we have to 'match' the syndromes into pairs.
- We base this decision on the probability of a chain connecting two syndromes, which we get from the error model.

There's an algorithm for this, called *minimum-weight perfect matching*, and it runs in polynomial ($\mathcal{O}(d^{6-12})$) time.

Known Decoders

‘Interleaved’ MWPM

The syndromes of the surface code appear at the ends of chains of errors:



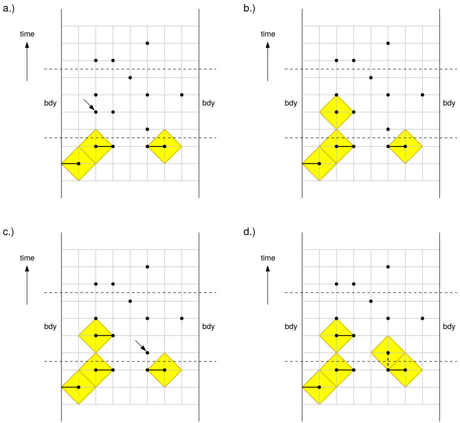
- To determine which error occurred, we have to ‘match’ the syndromes into pairs.
- We base this decision on the probability of a chain connecting two syndromes, which we get from the error model.

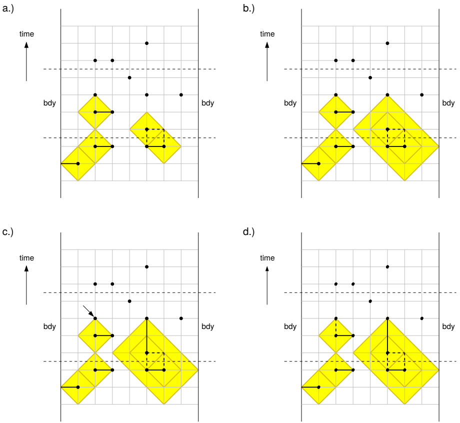
There’s an algorithm for this, called *minimum-weight perfect matching*, and it runs in polynomial ($\mathcal{O}(d^{6-12})$) time.
Our input graphs are extremely structured, though, so there are shortcuts we can take.

Known Decoders

‘Interleaved’ MWPM

To speed up the decoding, Fowler interleaves the running of the algorithm with the measurement of syndromes:





Unknown Decoders

Parallel Approximate MWPM

Approximating Minimum Weight Perfect Matchings for Complete Graphs Satisfying the Triangle Inequality

N.W. Holloway, S. Ravindran and A.M. Gibbons*

Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK.

Abstract. We describe an $O(log^3 n)$ time NC approximation algorithm for the CREW P-RAM, using $n^3/\log n$ processors with a $2\log_3 n$ performance ratio, for the problem of finding a minimum-weight perfect matching in complete graphs satisfying the triangle inequality. The algorithm is conceptually very simple and has a work measure within a factor of $log^2 n$ of the best exact sequential algorithm. This is the first NC approximation algorithm for the problem with a sub-linear performance ratio. As was the case in the development of sequential complexity theory, matching problems are on the boundary of what problems might ultimately be described as *tractable* for parallel computation. Future work in this area is likely to decide whether these ought to be regarded as those problems in NC or those problems in RNC.

- In 1990, A parallel algorithm was developed for approximating MWPM.

Unknown Decoders

Parallel Approximate MWPM

Approximating Minimum Weight Perfect Matchings for Complete Graphs Satisfying the Triangle Inequality

N.W. Holloway, S. Ravindran and A.M. Gibbons*

Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK.

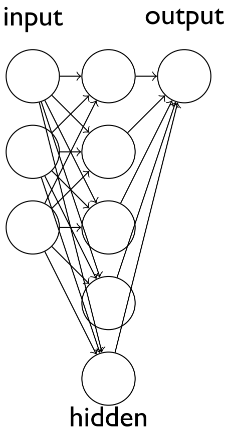
Abstract. We describe an $O(log^3 n)$ time NC approximation algorithm for the CREW P-RAM, using $n^3/\log n$ processors with a $2\log_3 n$ performance ratio, for the problem of finding a minimum-weight perfect matching in complete graphs satisfying the triangle inequality. The algorithm is conceptually very simple and has a work measure within a factor of $log^2 n$ of the best exact sequential algorithm. This is the first NC approximation algorithm for the problem with a sub-linear performance ratio. As was the case in the development of sequential complexity theory, matching problems are on the boundary of what problems might ultimately be described as *tractable* for parallel computation. Future work in this area is likely to decide whether these ought to be regarded as those problems in NC or those problems in RNC.

- In 1990, A parallel algorithm was developed for approximating MWPM.
- Nobody knows how well it works for decoding …

Unknown Decoders

Neural Networks

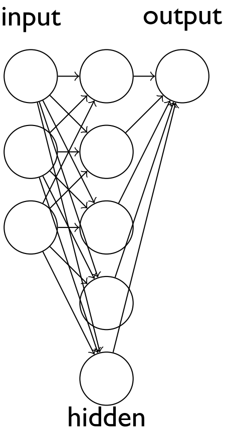
Neural networks are fitting functions that can be evaluated in parallel and optimized efficiently.



Unknown Decoders

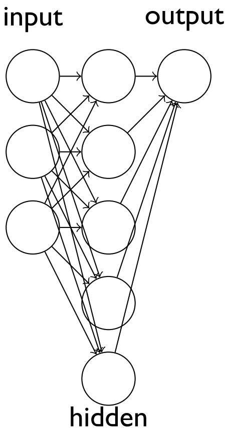
Neural Networks

Neural networks are fitting functions that can be evaluated in parallel and optimized efficiently.



- At every hidden/output vertex, evaluate $\frac{1}{1+\exp(\vec{w}\cdot\vec{x})}$ for input $\vec{x}$ and output result.

Neural networks are fitting functions that can be evaluated in parallel and optimized efficiently.

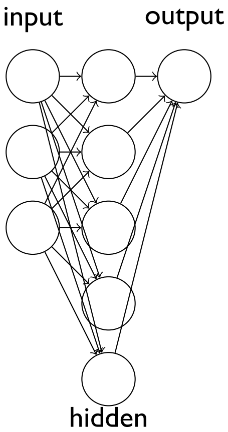


- At every hidden/output vertex, evaluate $\frac{1}{1+\exp(\vec{w}\cdot\vec{x})}$ for input $\vec{x}$ and output result.
- Composition implies gradient in $\vec{w}$-space can be simplified using the chain rule.

Unknown Decoders

Neural Networks

Neural networks are fitting functions that can be evaluated in parallel and optimized efficiently.

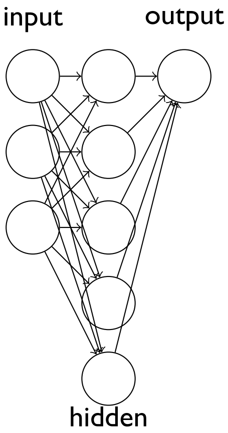


- At every hidden/output vertex, evaluate $\frac{1}{1+\exp(\vec{w}\cdot\vec{x})}$ for input $\vec{x}$ and output result.
- Composition implies gradient in $\vec{w}$-space can be simplified using the chain rule.
- For large enough networks, many knowledge-free fits can be obtained.

Unknown Decoders

Neural Networks

Neural networks are fitting functions that can be evaluated in parallel and optimized efficiently.



- At every hidden/output vertex, evaluate $\frac{1}{1+\exp(\vec{w}\cdot\vec{x})}$ for input $\vec{x}$ and output result.
- Composition implies gradient in $\vec{w}$-space can be simplified using the chain rule.
- For large enough networks, many knowledge-free fits can be obtained.
- Can they decode the surface code? (Our research.)

Unknown Decoders

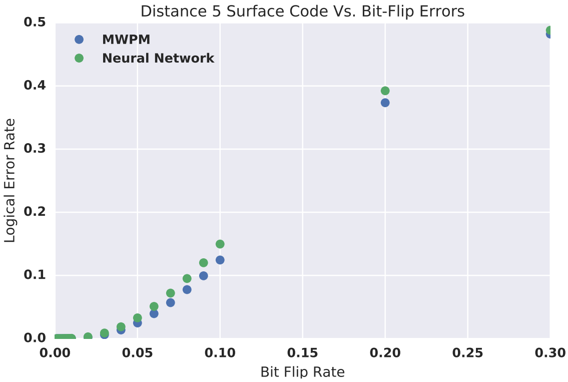
Neural Networks

The distance-5 code has 12-bit syndromes, and the decoder output is a single bit.

Unknown Decoders

Neural Networks

The distance-5 code has 12-bit syndromes, and the decoder output is a single bit. Using three hidden layers (8, 6, and 4 vertices), we can obtain semi-reasonable performance:



- Ongoing work with Savvas Varsamopoulos (TU Delft) and Xiaotong Ni (MPQ)
- Next steps: Encode space and time symmetries of decoding problem in neural network, move to higher distance.
- Central question: Can a threshold be obtained with a constant number of layers?

Questions?