

# WHAT ALGORITHMS SHOULD WE STUDY WITH 100 QUBITS AND 1M LOGICAL GATES?

**Alexandru Paler**

Aalto University, Helsinki, Finland  
Quantum Software and Algorithms Group

[alexandru.paler@aalto.fi](mailto:alexandru.paler@aalto.fi)

## Motivation: QC needs error-correction



Physical (raw) qubits

- not well behaved
- faulty - affected by environmental noise and manufacturing inconsistencies
- solitary (not many) on a device

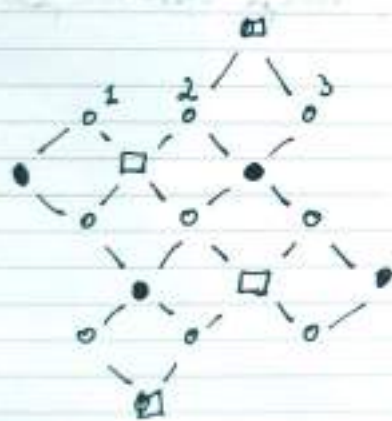
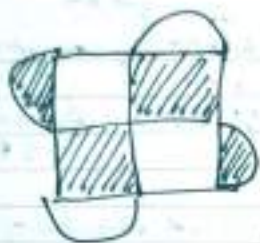


Error-corrected qubits

- controlling the risks
- not faulty - or controlled failure rates
- difficult to achieve due to lack of hardware qubits, not scalable classical software etc.

# **A Brief Introduction to Surface Codes**

# Surface Code



distance  
three

- - data qubit
- - synd X qubit
- - synd Z qubit

9 data q.

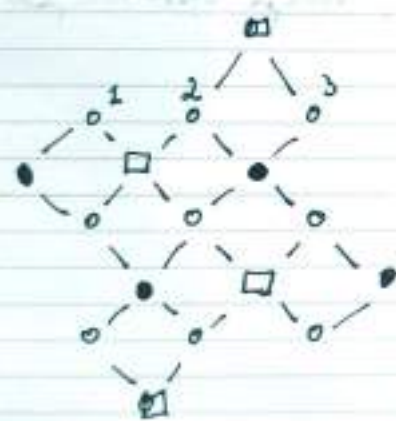
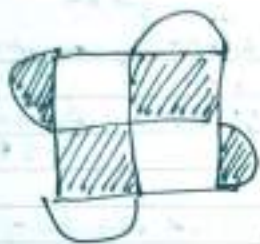
4 synd X

4 synd Z

---

17 qubits

## Surface Code



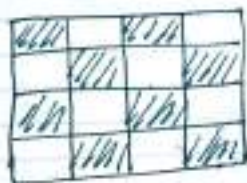
distance  
three

- - data qubit
- - synd X qubit
- - synd Z qubit

9 data q.  
4 synd X  
4 synd Z  
-----  
17 qubits

## Performing CNOTS

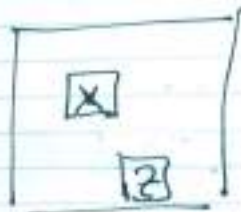
### Braiding



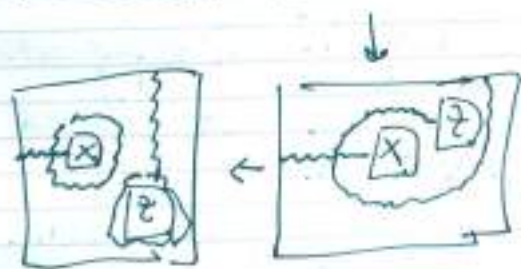
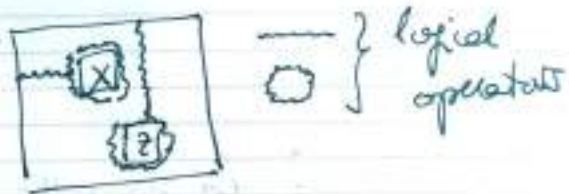
All plaquettes  
enforced.



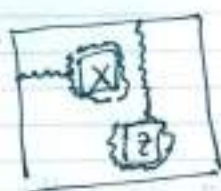
Two plaquettes  
not enforced  
two defects.



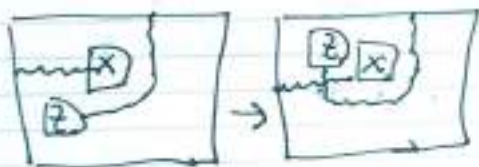
↓  
support for  
two logical qubits.



for more details  
arxiv. 1208.0928



 } logical operators



→



↓

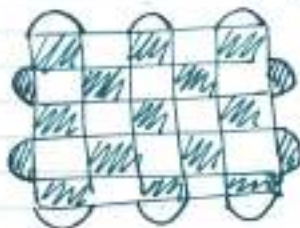


←



for more details  
arxiv. 1208.0928

## Lattice Surgery



$$D + Q = \text{shaded square}$$

$$D + D = \text{unshaded square}$$



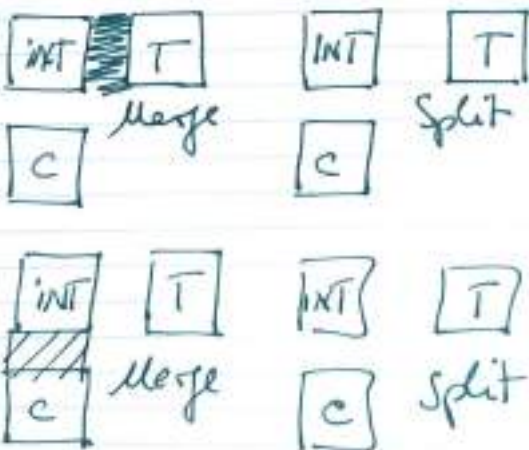
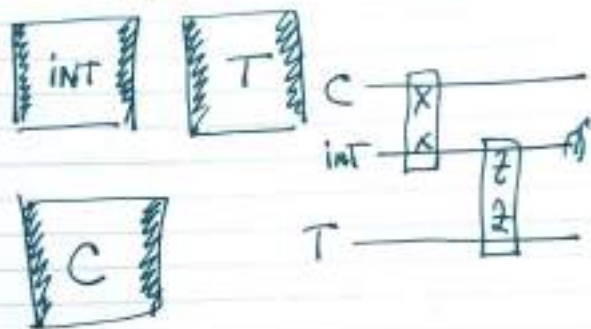
Merge



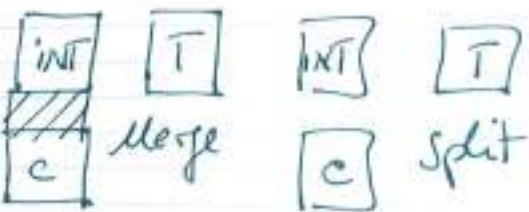
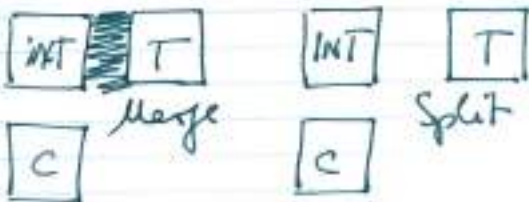
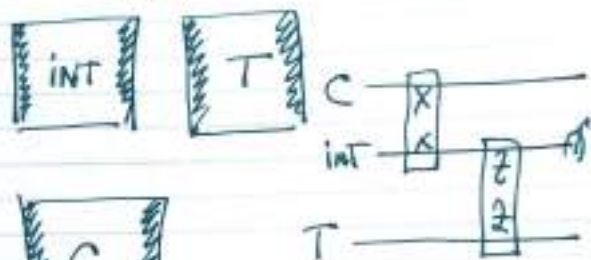
Split



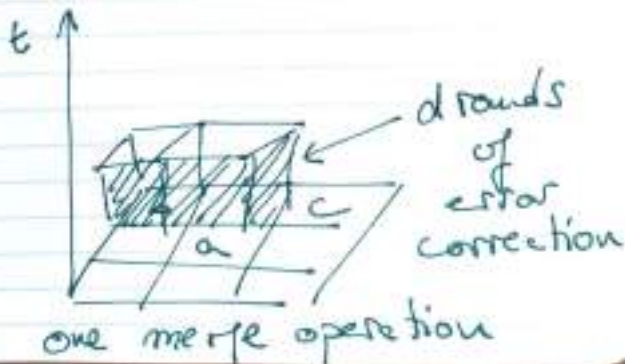
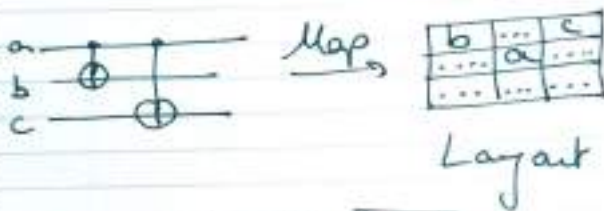
for more details  
arxiv 1111.4022



for more details  
arxiv 1111.4022

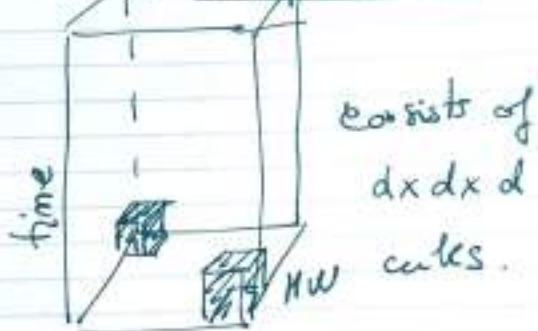


How are Circuits Compiled?





## Spatio-temporal Volume of a Computation.

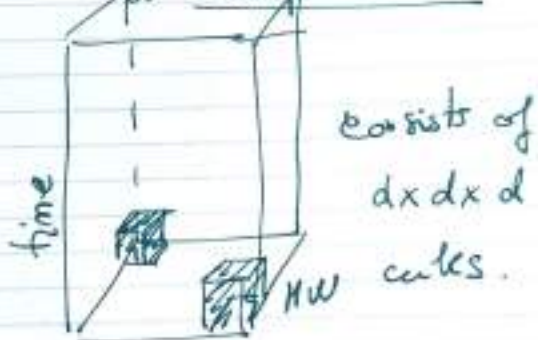


100 gates  $\approx$  200 patches

1 Million gates  $\approx 10^6$  ~~gates~~

$\Rightarrow 10^3$  volume

## Spatiotime Volume of a Computation.



100 qubits  $\approx$  200 patches

1 Million gates  $\approx 10^6$  ~~ops~~

$\Rightarrow 10^9$  volume

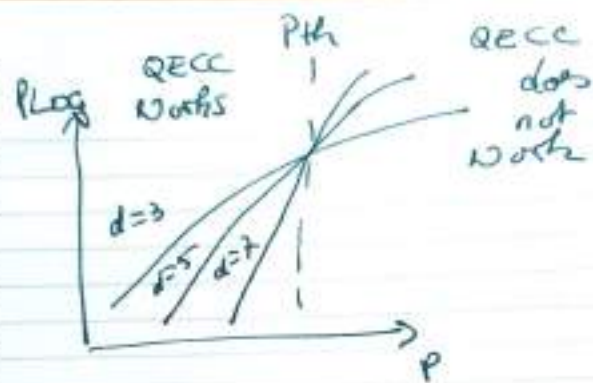
## Logical Error Rate.

prob failure  $\propto$  Volume.



Noise Model: single qubit  $\rho$   
two qubit 10p  
measurement 10p?

$$P_{\text{log}} \sim \left( \frac{P}{P_{\text{th}}} \right)^{\frac{d+1}{2}}$$



threshold prob.  $p_{th}$

$$P_{log} \sim \left( \frac{p}{p_{th}} \right)^{\frac{d+1}{2}}$$

Usual values:  $p \sim 0.1\%$

$p_{th} \sim 1\%$

increase  $d$  by 2  $\rightarrow$  10x lower  $P_{log}$

$d = 17$  min. req. for  $10^3$

# **Scalable (Machine Learning) Decoders**

# Faster Decoding by Pipelining and Parallelisation

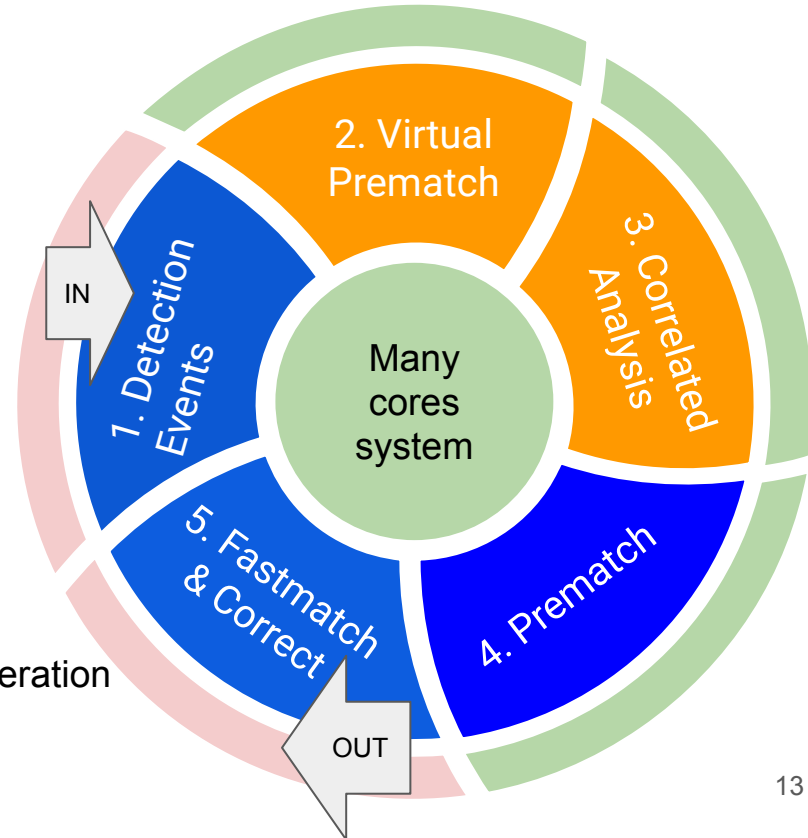
Goal: Real-time scalable decoding

Method:

1. Pipelining and Parallelism
2. Solve simple matches before full decoding
3. On-the-fly reweighting to account for error correlations
4. Do not sacrifice decoding performance for speed

Helper functionalities:

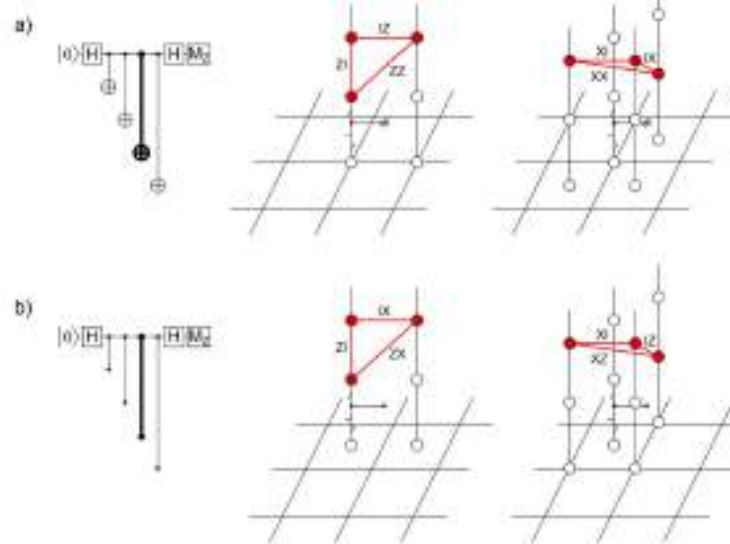
- a. Automatic computation of logical error rates
- b. Exhaustive test of single and double error correction
- c. Prepare, benchmark and validate 64 core real time operation
- d. Prepare for different code variants



# Correlated analysis of errors

Systematic to analyse and decompose correlated errors.

Circuit level simulation - Gates have a list of possible errors



# Correlated analysis of errors

Systematic to analyse and decompose correlated errors.

Circuit level simulation - Gates have a list of possible errors

Errors generate detection events: 1 (connect to boundary), 2, 2+

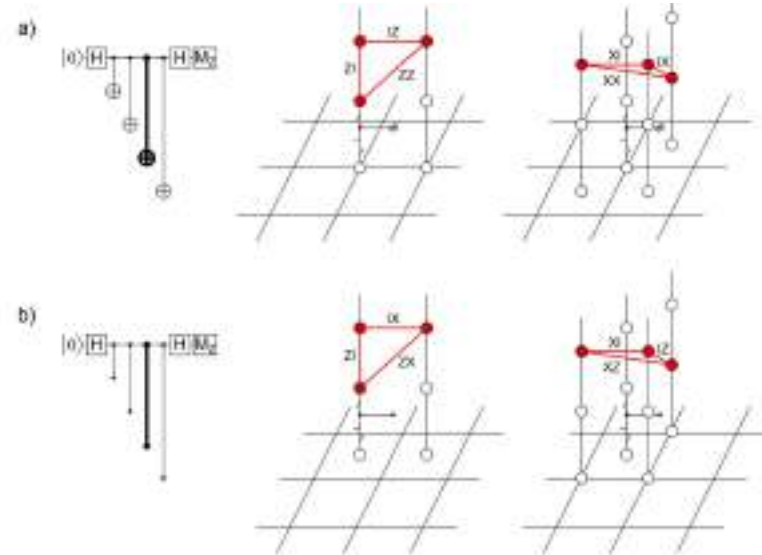


2+ Detection events from a single error have to be decomposed

Treat 2+ events like hyperedges: decompose into known edges

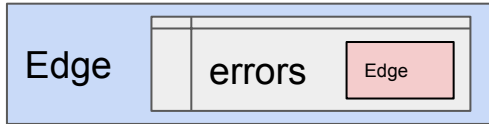


An edge is generated by errors, the probability of an edge is determined by the prob of the errors



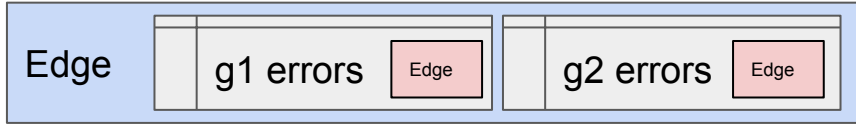
# Correlated analysis of errors

An edge is generated by errors,  
the probability of an edge is determined by the prob of the errors



Errors are generated by gates.

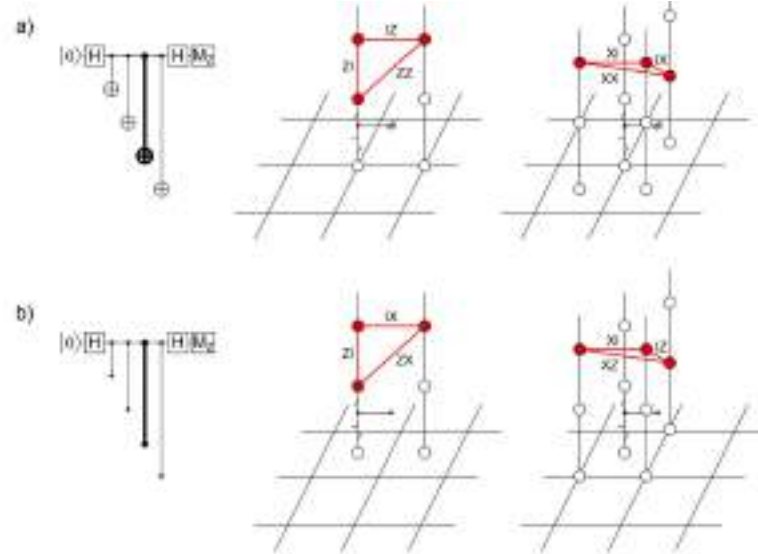
Assume independent gate errors.



$p_{\text{corr}}$  = sum of probs from gate errors.

Update edge probability (after prematching)

$p_{\text{new}} = p_{\text{old}} + p_{\text{corr}}$





# Results: pipelining and correlated decoding

Fast decoding with comparable results to arXiv:1310.0863v1  $d_3@10^{-5} < 10^{-7}$   $d_9@10^{-3} < 10^{-7}$

Hardware agnostic correlated analysis for reweighting edges

Prematching to support reweighting and fastmatch

Pipelined correlated minimum weight perfect matching of the surface code

Alexandru Paler<sup>1,2</sup> and Austin G. Fowler<sup>3</sup>

<sup>1</sup>Aalto University, Espoo 00150, Finland

<sup>2</sup>University of Texas at Dallas, Richardson, TX 75080, USA

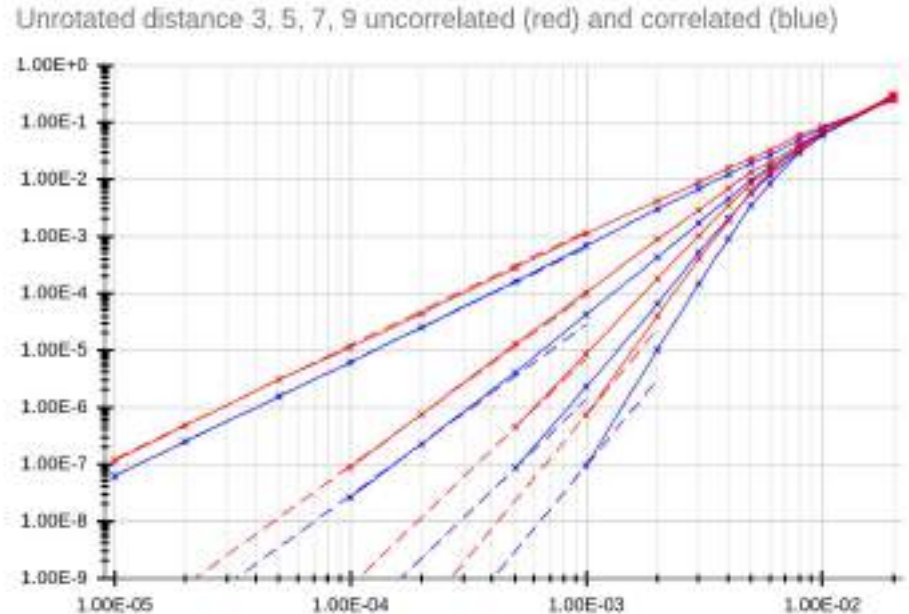
<sup>3</sup>Google Inc., Santa Barbara, 93117 CA, USA

Published: 2020-12-12, volume 7, page 1203

arXiv: 2009.09628v2

DOI: <https://doi.org/10.22331/qj-2020-12-12-1203>

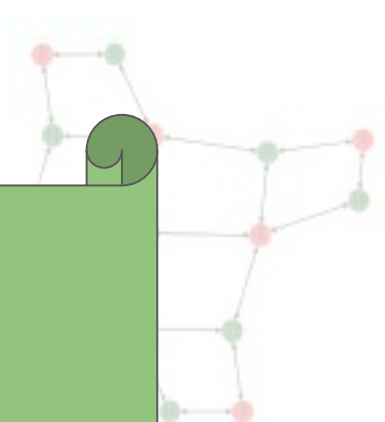
Quantum 7, 1203 (2023)



# ML Decoders: Introduction and Motivation

Optimal Decoding of QECC is a hard problem [1]

Belief propagation (BP) - one of the best-known classical decoding algorithms



We want to build a NN based decoder

- **which is learning fast and which can operate fast**
- works for LDPC codes – also the surface code

**We present a decoder that is learning the constraints of QECC decoding**

Neural network (NN) decoding has constant decoding runtime 🗨️ 😊

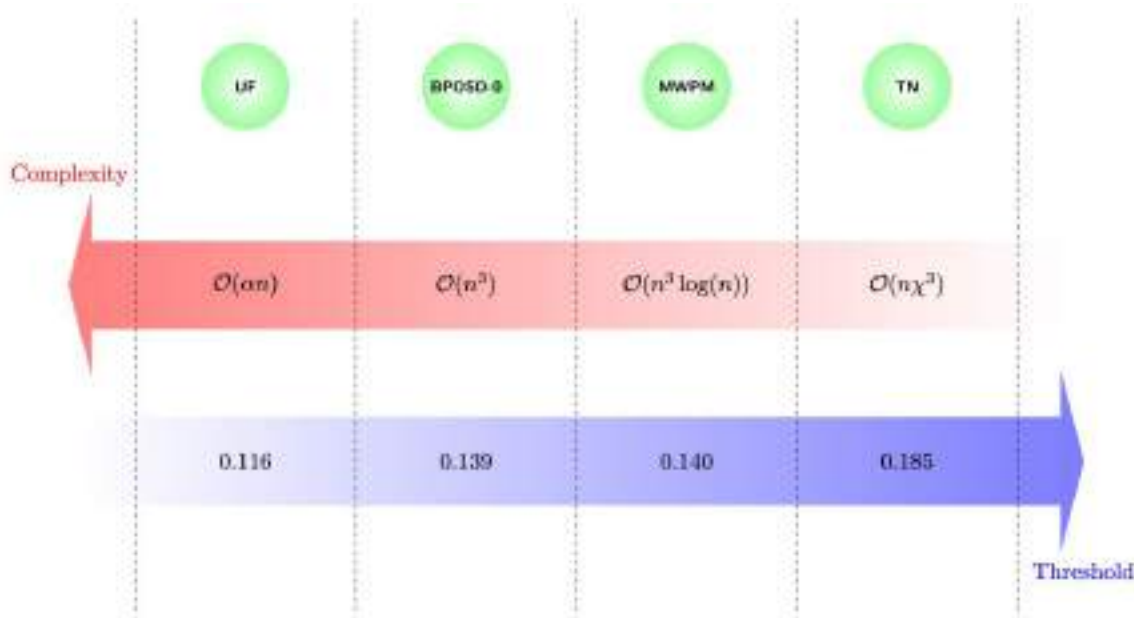
Limitations of previous NN based decoding approaches:

- Different NN architectures for different code types
- Retain for each code distance
- there is a GNN decoder [4], but it does not work like we want it

- [1] <https://arxiv.org/abs/1310.3235>
- [2] <https://arxiv.org/abs/1811.07835>
- [3] <https://arxiv.org/abs/2212.03214>
- [4] <https://arxiv.org/abs/2307.01241>
- [5] <https://arxiv.org/abs/2005.07016>

# Why ML Decoders?

ML Decoding has linear time (although the scaling of the models with code distance is not known)



iOlius, A. D., Fuentes, P., Orús, R., Crespo, P. M., & Martinez, J. E. (2023). Decoding algorithms for surface codes. *arXiv preprint arXiv:2307.14989*.

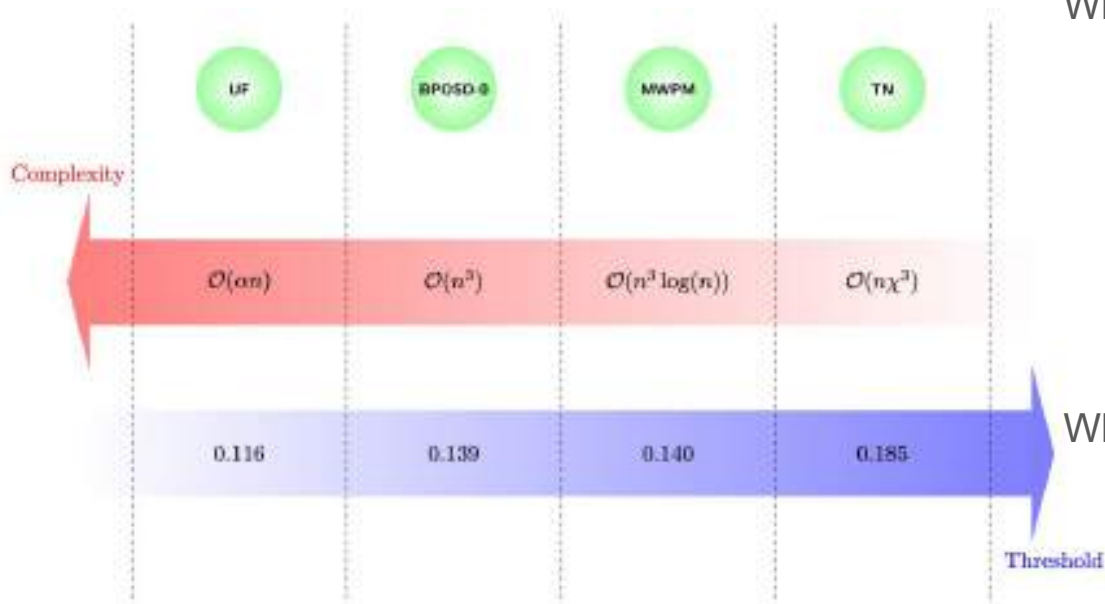
# Why ML Decoders?

ML Decoding has linear time (although the scaling of the models with code distance is not known)

What the goal is:

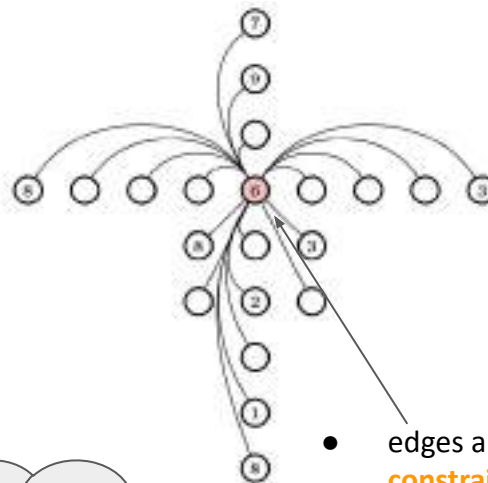


What the state of the art is:



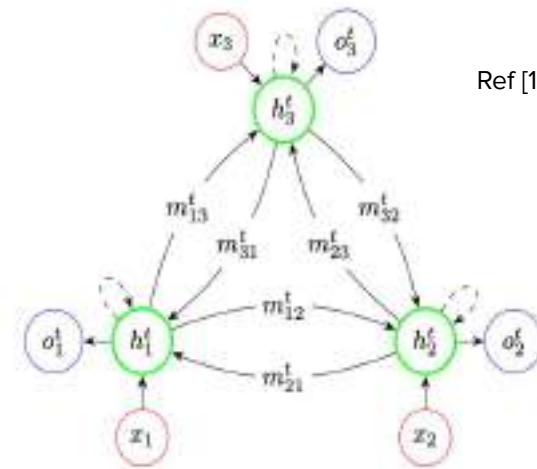
# The Graph Neural Network (GNN) Decoder Learning BP to Satisfy Constraints

5	3		7				
6			1	9	5		
	9	8				6	
8			6			3	
4		8		3		1	
7			2			6	
	6				2	8	
			4	1	9	5	
			8			7	9



- edges are **constraints** necessary for the solution
- vertices are forming constraint pairs

Decoding works like solving Sudoku – solve the **constraints**



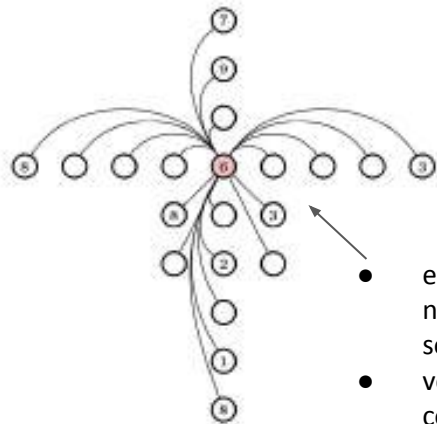
Ref [1]

**red:** input vertices in GNN  
**blue:** output  
**green:** node state  
messages are sent along the edges

# Graph Neural Network (GNN) Decoder

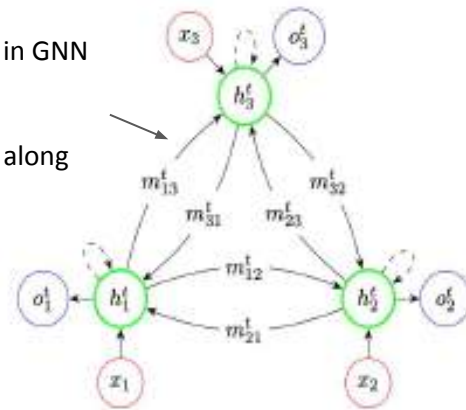
## The Sudoku analogy - Learning BP

5	3		7					
6			1	9	5			
	9	8					6	
8			6					3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



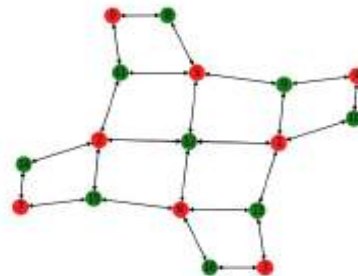
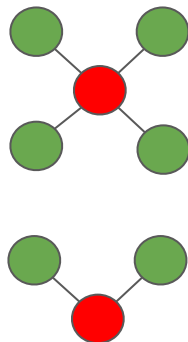
- edges are **constraints** necessary for the solution
- vertices are forming constraint pairs

**red**: input vertices in GNN  
**blue**: output  
**green**: node state  
 messages are sent along the edges



?		?		?
	0		1	
?		?		?
	1		0	

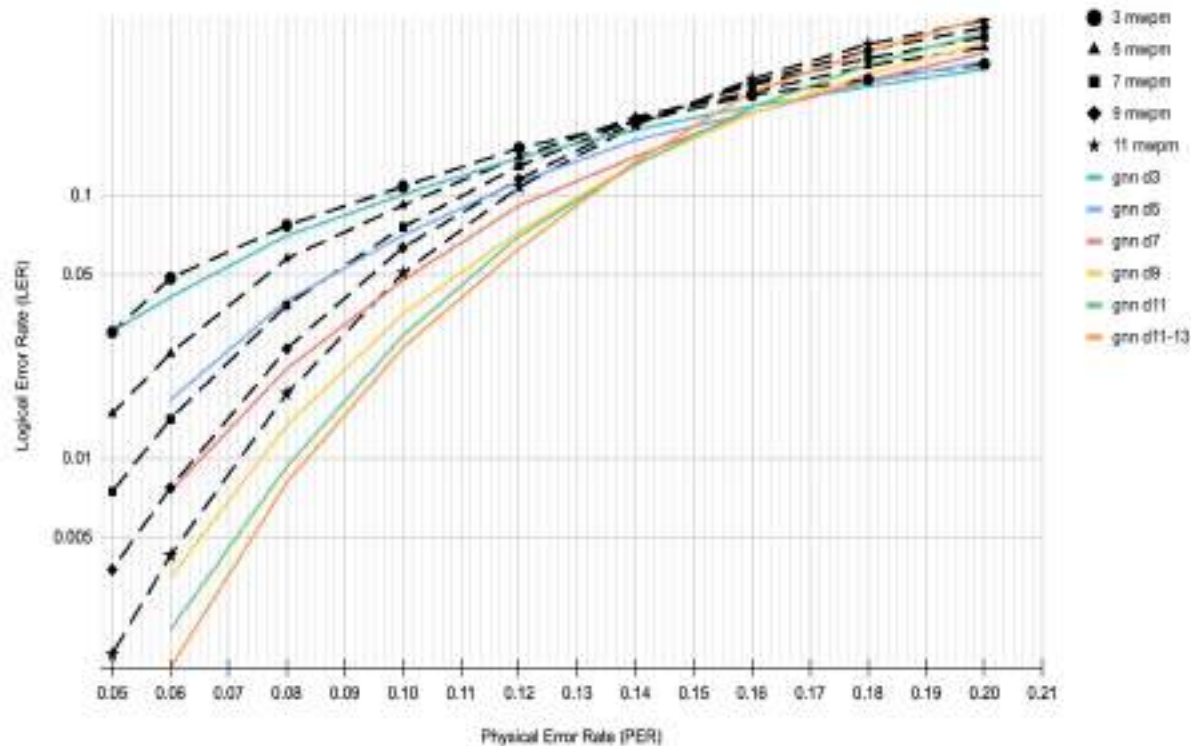
**Red** = filled values = syndromes  
**Green** = to fill = errors / data qubits



Tanner graph for surface code of distance 3: **RED** vertices are check nodes, **GREEN** vertices are data nodes

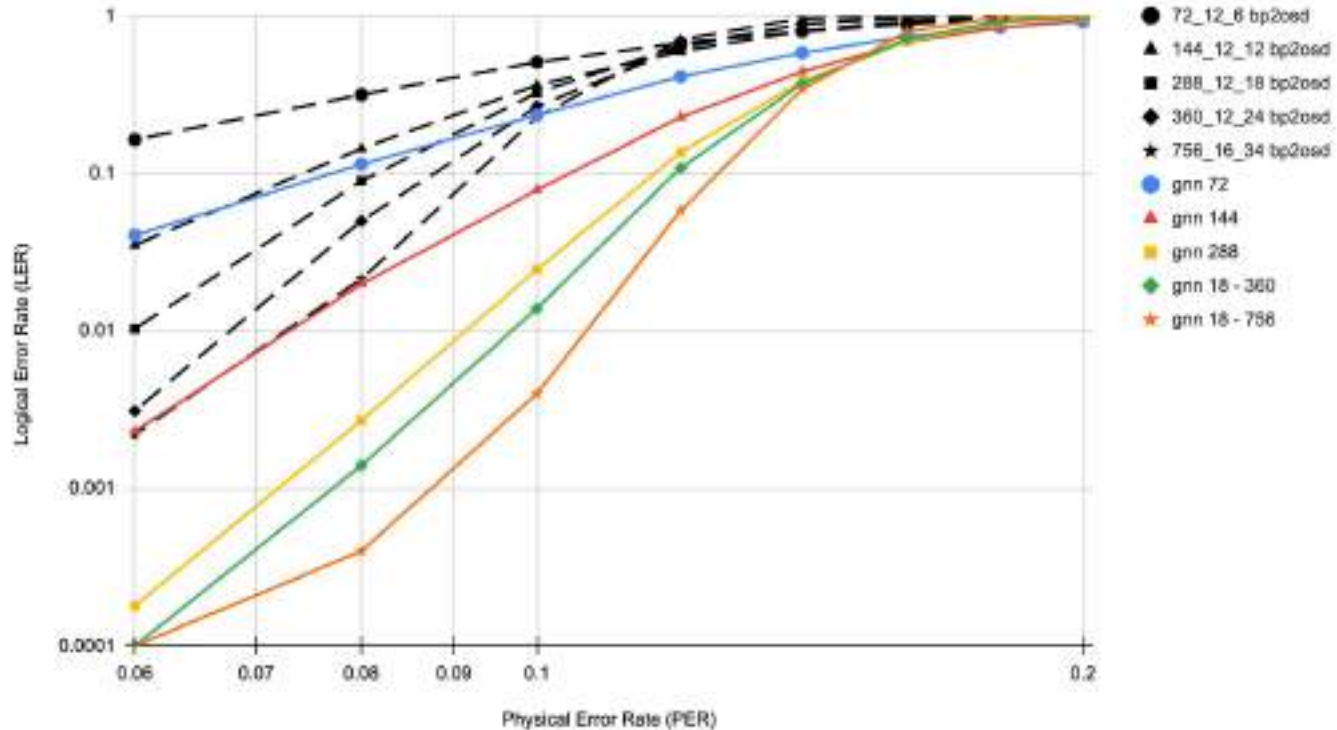
# GNN as replacement of MWPM or BPOSD for Surface code

GNN vs MWPM



# GNN as replacement of BP2-OSD for IBM's BB code

gnn vs bp2osd





# Very Fast Compilers (for Lattice Surgery)



<https://github.com/latticesurgery-com/>

Unitary  
Fund

SFU

SIMON FRASER  
UNIVERSITY

A"  
Aalto University

**Our Challenge:** *Logical Computations at scale*  
*100s to 1000s of logical qubits*

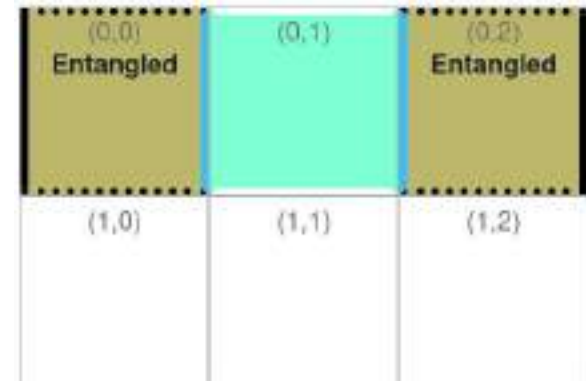
- Start with a lattice of NN connected qubits that can operate a Surface Code Cycle
- This lattice is partitioned into **tiles**.
- A tile can hold a **patch**, which encodes a logical qubit in a planar code
- Patches have different kinds of **boundaries** that are used to perform multibody measurements
- Unused lattice can be used as **routing** to carry out measurements among patches with no shared boundary

arXiv:2302.02459 (quant-ph)

(Submitted on 5 Feb 2023)

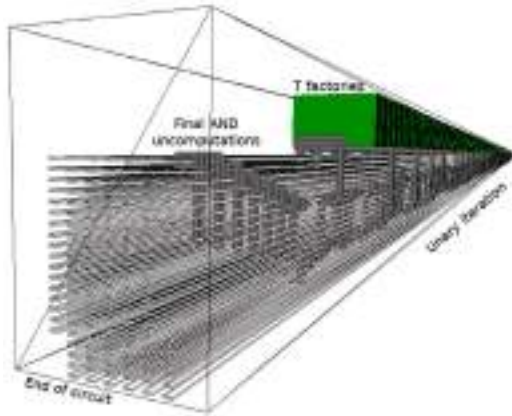
**A High Performance Compiler for Very Large Scale Surface Code Computations**

George Watkins, Hoang Minh Nguyen, Varun Seshadri, Keelan Watkins, Steven Pearce, Hoi-Kwan Lau, Alexandru Pater



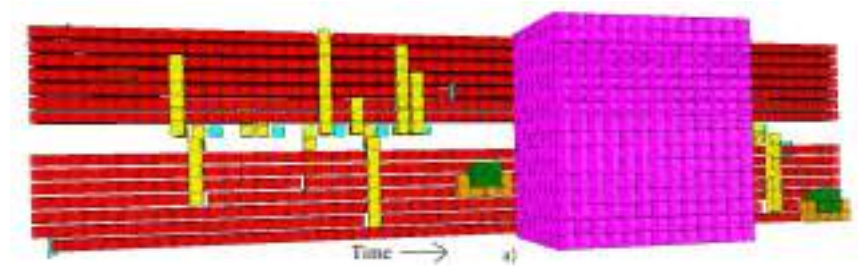


# Scalable Reinforcement Learning



Space-time volume of braided circuit

Babbush, Ryan, et al. "Encoding electronic spectra in quantum circuits with linear T complexity." *Physical Review X* 8.4 (2018): 041015.



Space-time volume of lattice surgery (LS) circuit

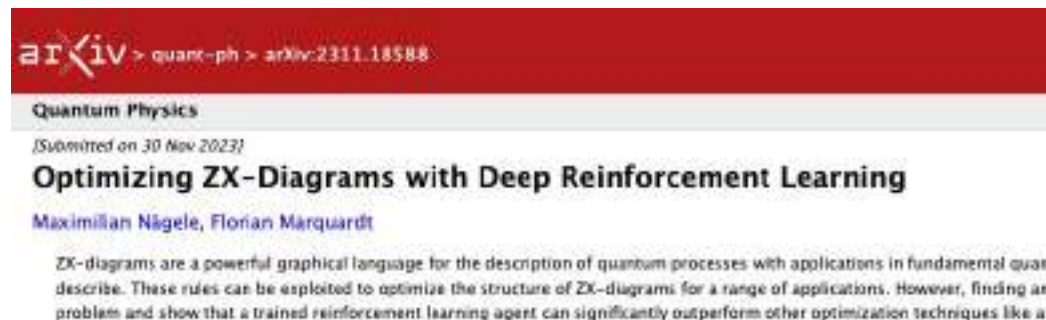
Paler, Alexandru, and Austin G. Fowler. "OpenSurgery for topological assemblies." *2020 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2020.

How do we optimize this? Maybe we can discover optimization algorithms...Let's use RL....

# RL Scalability

Training the agent takes around 41 hours on a single compute node with 32 CPUs and 2 GPUs. We run multiple environments in parallel on the CPUs during the sampling phase and train the agent distributed on both GPUs. The implementation of the algorithm could directly take advantage of larger compute nodes to speed up training time.

As AI applications demand greater compute power, efficiency may be improved via better chip design. The Nature paper [1] by Google researchers, published in June 2020, was advertised as a chip-design breakthrough using Machine Learning (ML). It addressed a challenging problem to optimize locations of circuit components on a chip and described applications to five TPU chip blocks, implying that no better methods were available at the time in academia or industry. The paper generalized the claims beyond chip design to suggest that Reinforcement Learning (RL) outperforms state of the art in combinatorial optimization.



arXiv > quant-ph > arXiv:2311.18588

Quantum Physics

(Submitted on 30 Nov 2023)

### Optimizing ZX-Diagrams with Deep Reinforcement Learning

Maximilian Nägele, Florian Marquardt

ZX-diagrams are a powerful graphical language for the description of quantum processes with applications in fundamental quantum computing. These rules can be exploited to optimize the structure of ZX-diagrams for a range of applications. However, finding an optimal structure is a challenging problem. In this paper, we demonstrate a problem and show that a trained reinforcement learning agent can significantly outperform other optimization techniques like a



arXiv > cs > arXiv:2306.09633

Computer Science > Machine Learning

(Submitted on 16 Jun 2023 (v1); last revised 29 Sep 2023 (this version, v2))

### The False Dawn: Reevaluating Google's Reinforcement Learning for Chip Macro Placement

Igor L. Markov

Reinforcement learning (RL) for physical design of silicon chips in a Google 2021 Nature paper stirred controversy due to poorly documented claims that raised eyebrows. This paper demonstrates that Google RL lags behind (i) human designers, (ii) a well-known algorithm (Simulated Annealing), and (iii) generally-available commercial software, while also reporting. Before publishing, Google rebuffed internal allegations of fraud. We note policy implications and conclusions for chip design.



**Fast & scalable RL is needed**

# RL Introduction

**Reinforcement Learning:** *trial and error* approach iteratively

**Goal:** *finding the optimal policy that solves a specific problem*

**Quantum circuit optimization:** reduce **cost** of the circuit

**Cost** can be many things:

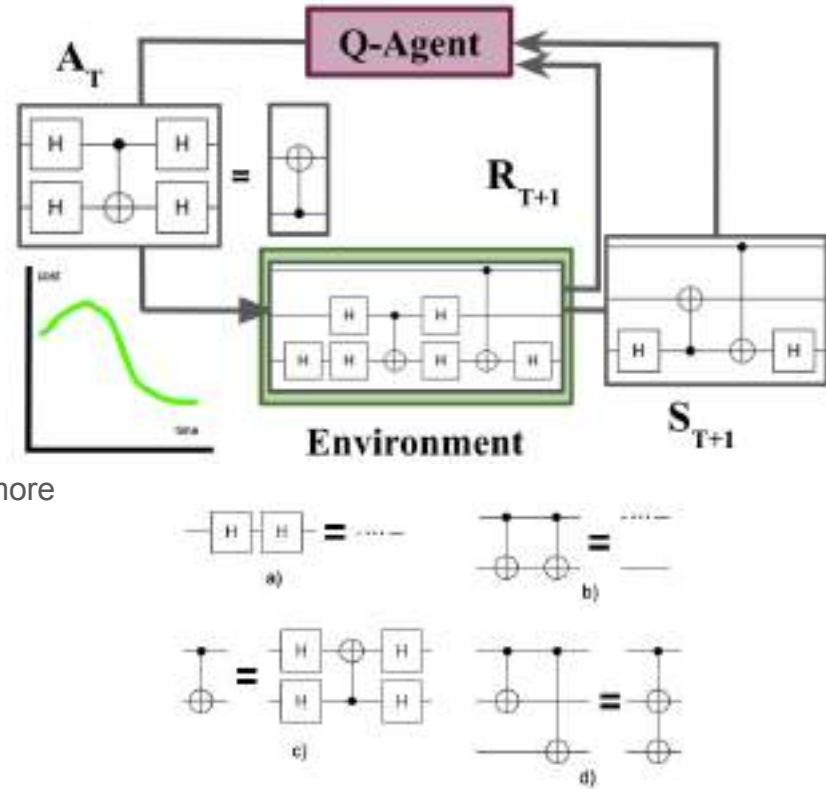
- Number of gates and their specific cost (e.g two-qubit gates are more expensive)
- Depth of the circuit
- Number of qubits
- How well does the circuit map to a qubit layout

**Our goal:** *use RL to reduce the cost of the circuit*

**Problem:** rewrite-rule based optimisation tends to increase the depth before it reaches optimum



**Difficult optimization landscape**



# Quantum Circuit Optimization Landscape

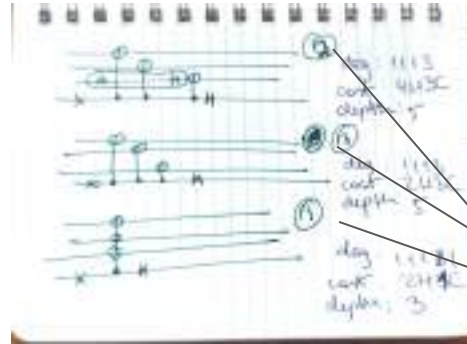


**2. But the path reaches summit first**

$$\text{reward}_T(C) = \left( \frac{\theta_T^0}{\text{degree}(C)} + \frac{\theta_T^1}{\text{cost}(C)} \right)^{\frac{\theta_T^2}{\text{depth}(C)}}$$

**3. And exploring too much might kill you...**

# Example of a RL Epoch

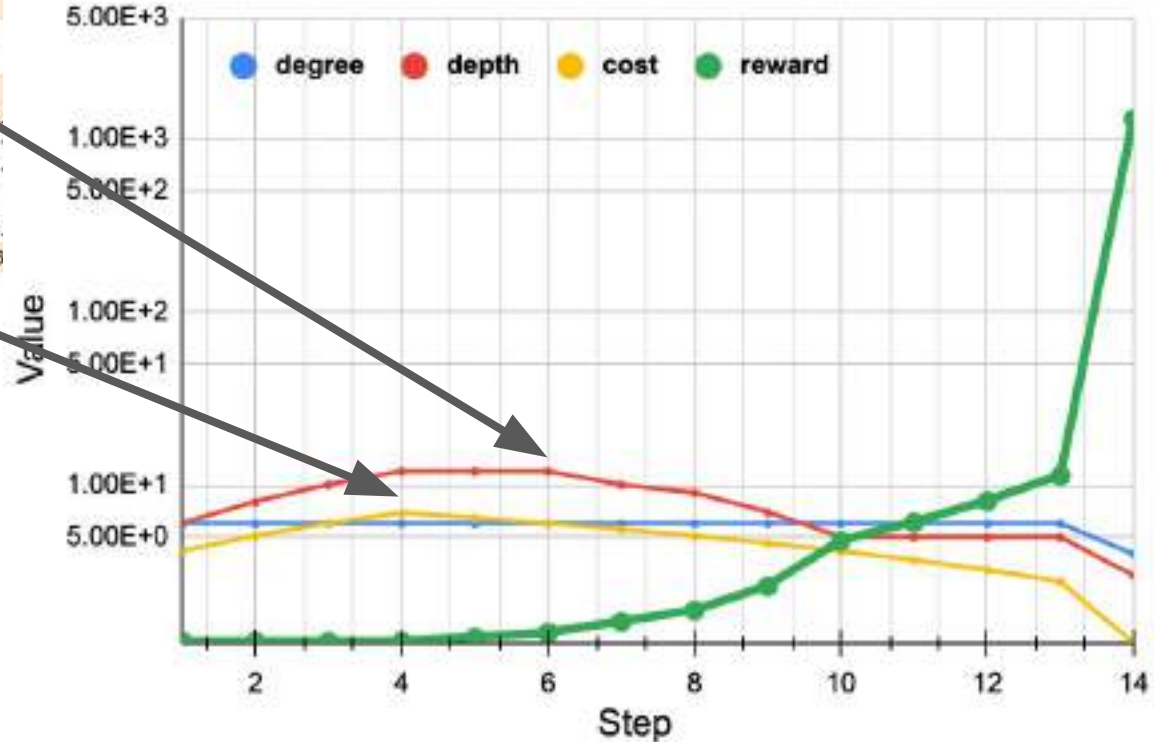


circuit	1	2	3	4	5	6	7	8	9	10	11	12	13	14
degree	6	6	6	6	6	6	6	6	6	6	6	6	6	4
scount	8	12	16	20	18	16	14	12	10	8	6	4	2	2
mcount	3	3	3	3	3	3	3	3	3	3	3	3	3	1
depth	6	8	10	12	12	12	10	9	7	5	5	5	5	3
cost	4.15	5.08	6.00	6.92	6.46	6.00	5.54	5.08	4.62	4.15	3.69	3.23	2.77	1.23
avg degree	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1
maxavgdegree	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
maxdepth	6	8	10	12	12	12	12	12	12	12	12	12	12	12
maxcost	4.15	5.08	6.00	6.92	6.92	6.92	6.92	6.92	6.92	6.92	6.92	6.92	6.92	6.92
reward	1.250	1.250	1.250	1.250	1.321	1.404	1.627	1.893	2.610	4.766	6.105	8.117	11.334	1296.000



# Example of a RL Epoch

circuit	1	2	3	4	5	6	7	8	9	10	11	12	13	14
degree	6	6	6	6	6	6								
scount	8	12	16	20	18	16								
mcount	3	3	3	3	3	3								
depth	6	8	10	12	12	12								
cost	4.15	5.08	6.00	6.92	6.46	6.00								
avg degree	1.5	1.5	1.5	1.5	1.5	1.5								
maxavgdegree	1.5	1.5	1.5	1.5	1.5	1.5								
maxdepth	6	8	10	12	12	12								
maxcost	4.15	5.08	6.00	6.92	6.92	6.92								
reward	1.250	1.250	1.250	1.250	1.391	1.404								



by recording the maximum, future rewards can only get better:

- minor improvements if close to maximum
- huge improvements if far from the maximum

$$reward_T(C) = \left( \frac{\theta_T^0}{degree(C)} + \frac{\theta_T^1}{cost(C)} \right) \frac{\theta_T^2}{depth(C)}$$

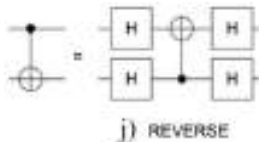
# Very Large Scale Circuit Optimizer

# Motivation

No software can handle gate optimization in **randomly** chosen circuit locations for circuits with *millions (billions?)* of gates!

Optimizer	Time
Cirq 1.2.0	> 20 hours
Tket 1.21.0	~ 1 min
PostgreSQL 14	?

Benchmarked state-of-the-art optimizers with circuits of 1 million templates.



Why **random**? ➔ circuit optimisation is a combinatorial (not sequential) problem. In-memory optimizers **are slow** for random memory access! Databases **are faster**.

TABLE IV. Resources required for quantum simulation of a planar Hubbard model with periodic boundary conditions and spin, as in Eq. (56). The dimension of the system indicates how many sites (spatial orbitals) are on each side of the square model. The number of system qubits is thus twice the number of spatial orbitals. The number of logical ancillae is computed as Eq. (6). Finally, the number of T gates is computed using Eq. (63), which assumes that  $u/t = 4$  and  $\Delta E = t/100$ . The first three problem sizes in the table are near the classically intractable regime.

Dimension	Spin orbitals	Logical ancilla	Total logical	T count
$6 \times 6$	72	33	105	$9.3 \times 10^3$
$8 \times 8$	128	33	161	$2.9 \times 10^8$
$10 \times 10$	200	36	236	$7.1 \times 10^8$
$20 \times 20$	800	42	842	$1.2 \times 10^{10}$

Example of practical circuit sizes

Encoding Electronic Spectra in Quantum Circuits with Linear T Complexity

Ryan Babbush, Craig Gidney, Dominic M. Berry, Nathan Wiebe, Jarrod McClean, Alessandro Perera, Austin Fowler, and Hartmut Neven  
 Phys. Rev. X **8**, 041048 – Published 23 October 2018

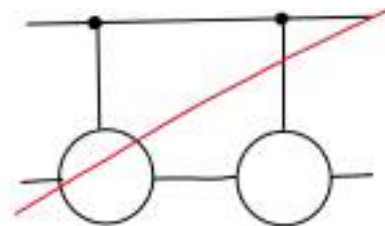
# Methods

We consider four types of gate templates:

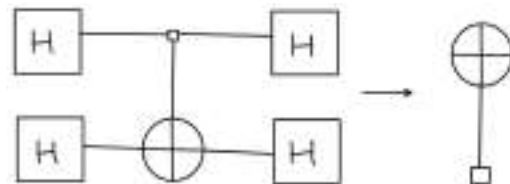
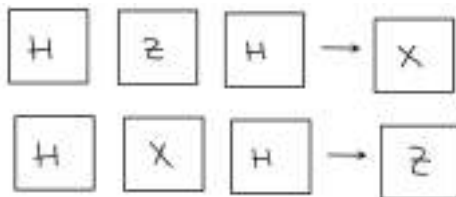
- **Single-qubit gate cancellations**



- **Two-qubit gate cancellations**



- **Base changes**



- **Commutations**

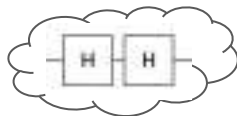


# Results: Random Synthetic Circuits

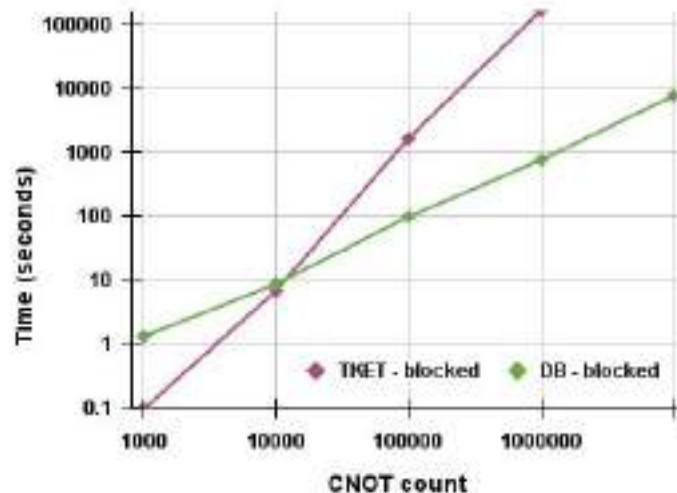
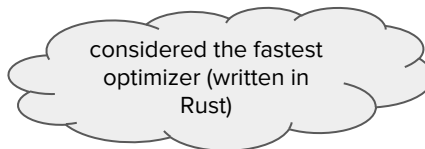
## Generating Synthetic Benchmark Circuits

1. Start from empty circuit - identity on all qubits
2. For `nr` in range(`LARGE_NUMBER`)
  - a. Select random qubit(s)
  - b. Insert pairs of cancelling gates
    - i. Hadamard gates
    - ii. CNOTs

e.g. `LARGE_NUMBER` = 1 million (see next slides)

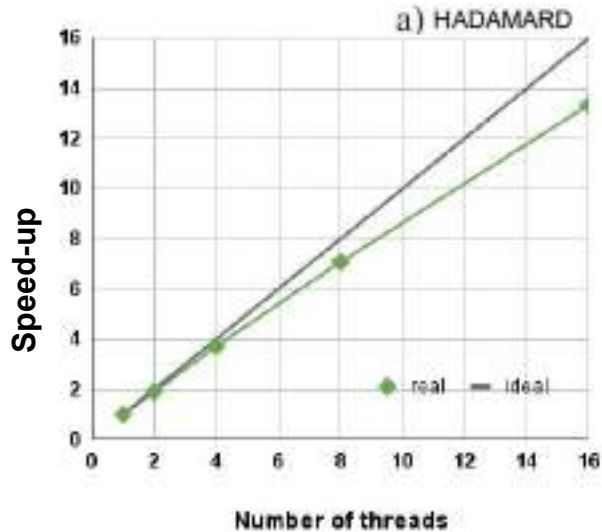
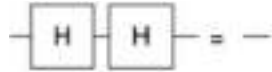


- Our tool is faster than `tket`.
  - for more than 10k gates
  - speed-up increases with circuit size

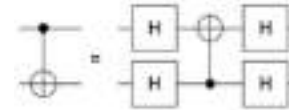


# Results: Multi-threaded performance

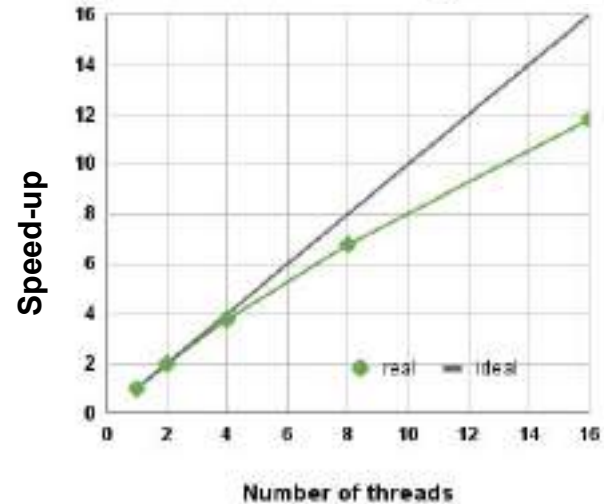
Type-1



Type-2



j) REVERSE



Our benchmark circuit contains 1 million templates of either **Type-1** or **Type-2**

- 2 million gates when using type-1
- 5 million gates when using type-2

# Conclusion: Executing algorithms/circuits of 100 qubits and 1M gates requires more work

## 1. Decoders

- a. Non-ML Decoders can be sped up by pipelining and parallelization  
<https://arxiv.org/abs/2205.09828>
- b. GNN Decoders seem to be learning the messages and algorithms of a message passing

## 2. Large scale compilation and optimization

- a. (Reinforcement) Learning of optimization algorithms has many bottlenecks
- b. Engineering Reward Functions seems to speed/improve RL <https://arxiv.org/abs/2311.12498>
- c. Compression of RL states with autoencoders <https://arxiv.org/abs/2303.03280>
- d. Parallelization and Fast Random Access of circuit optimization
  - i. ensures correctness of rewrites performed in parallel on the circuit
  - ii. supports analysis of circuits (e.g. average number of CNOT gates between pairs of neighbouring T gates)