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

Arshpreet Singh Maan
Ioana Moflic, Alexandru Paler

Aalto University, Helsinki, Finland
Quantum Software and Algorithms Group

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

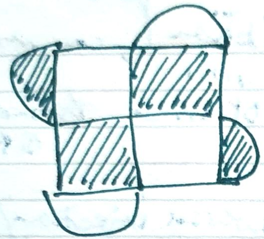
- controlling the risks
- not faulty - or controlled failure rates
- difficult to achieve due to lack of hardware qubits, not scalable classical software etc.

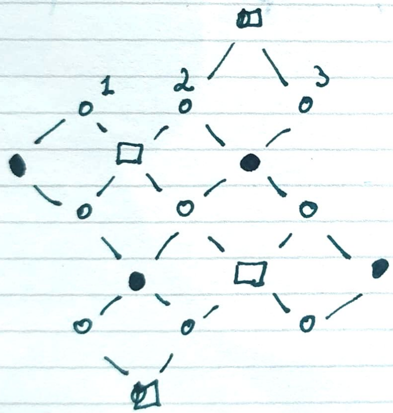
funded by: DARPA QUANTERA ERA-NET Cofund in Quantum Technologies Google MINISTRY OF EDUCATION AND CULTURE

Aalto University

A Brief Introduction to Surface Codes

Surface Code





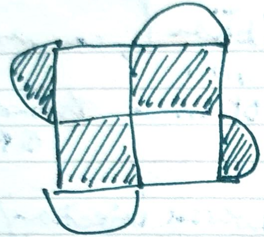
distance three

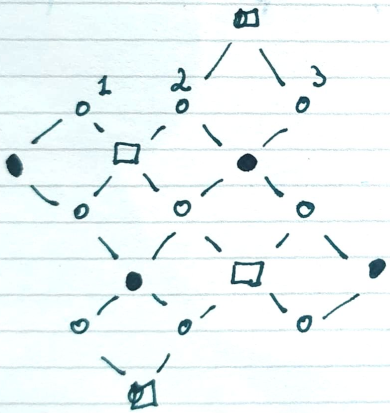
9 data q.
4 synd X
4 synd Z

17 qubits

○ - data qubit
● - synd X qubit
□ - synd Z qubit

Surface Code





distance three

○ - data qubit

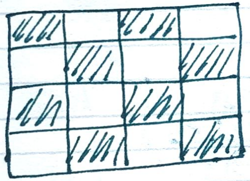
● - synd X qubit

□ - synd Z qubit

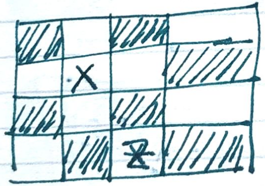
9 data q.
4 synd X
4 synd Z
17 qubits

Performing CNOTS

Braiding



All plaquettes enforced.



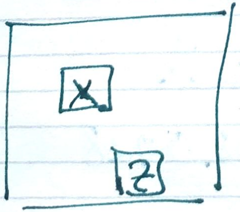
Two plaquetts not enforced

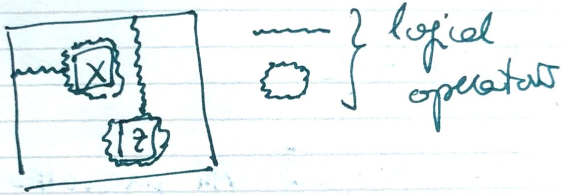
=

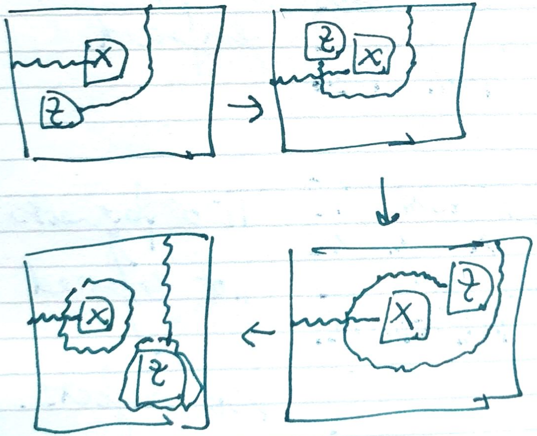
two defects.

⇓

support for two logical qub.

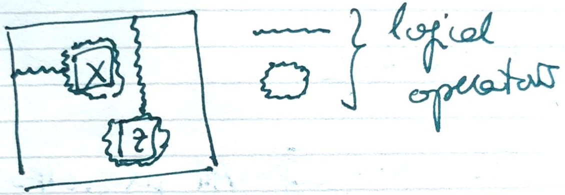


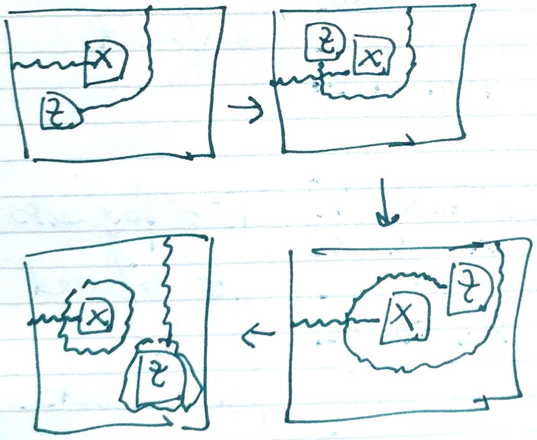




for more details

arxiv. 1208.0928

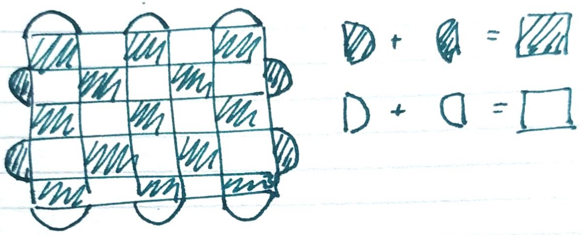


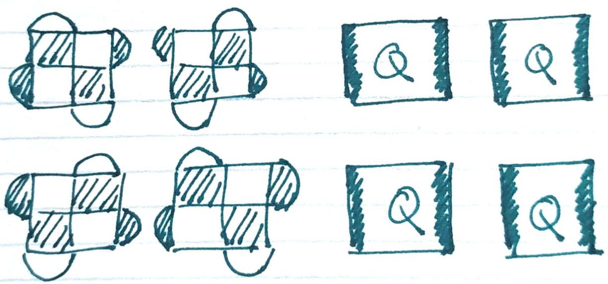


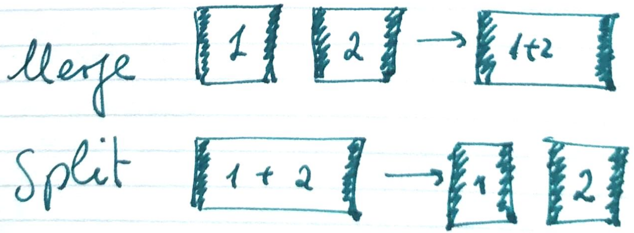
for more details

arxiv. 1208.0928

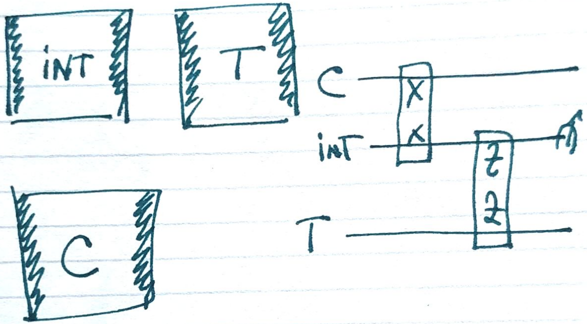
Lattice Surgery

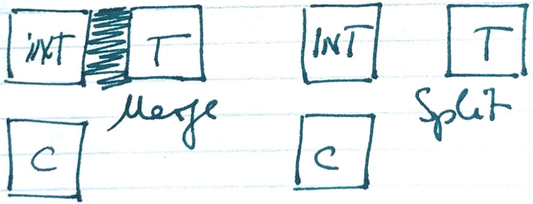


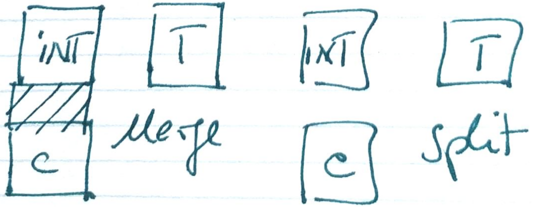




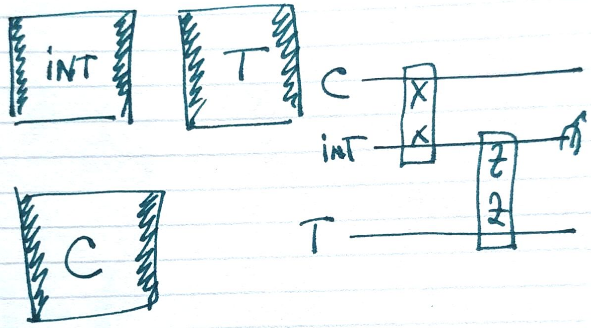
for more details
arxiv 1111.4022

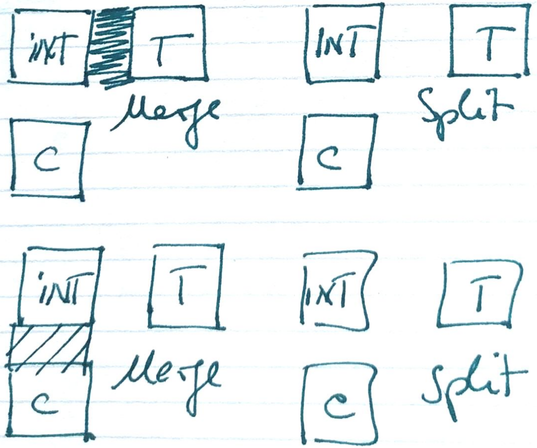




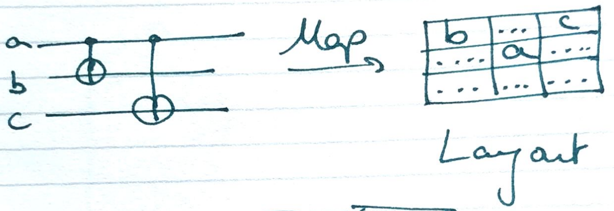


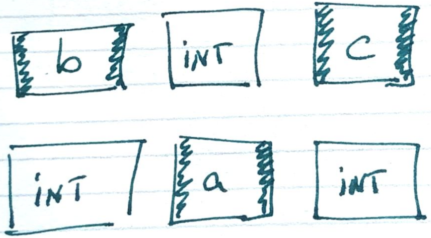
for more details
arxiv 1111.4022

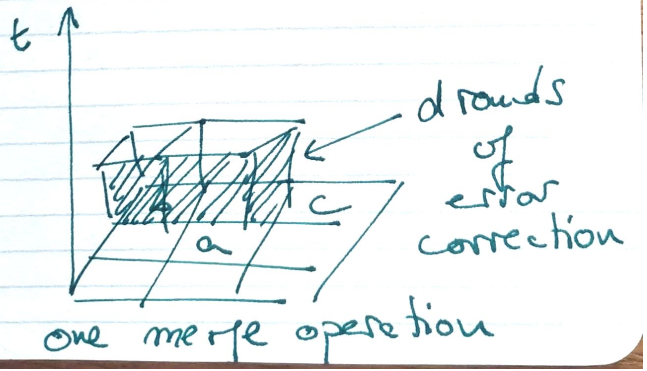




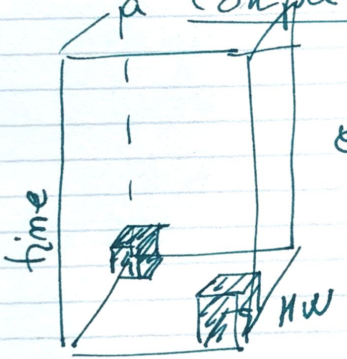
How Are Circuits Compiled?







Spacetime Volume of a Computation.



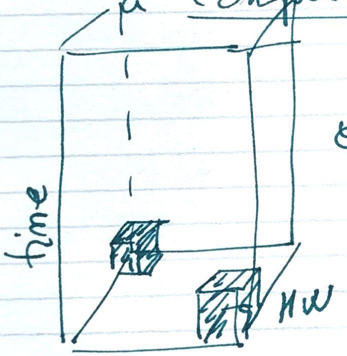
consists of $d \times d \times d$ cubes.

100 qubits ⇝ 200 patches

1 Million gates ~ $10^6 \cdot 2$

$\Rightarrow$ 10^9 volume

Spacetime Volume of a Computation.



consists of $d \times d \times d$ cubes.

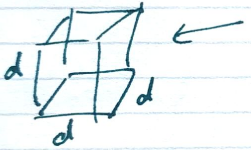
100 qubits ⤳ 200 patches

1 Million gates ~ $10^6 \cdot 2$

⇒ 10^9 Volume

Logical Error Rate.

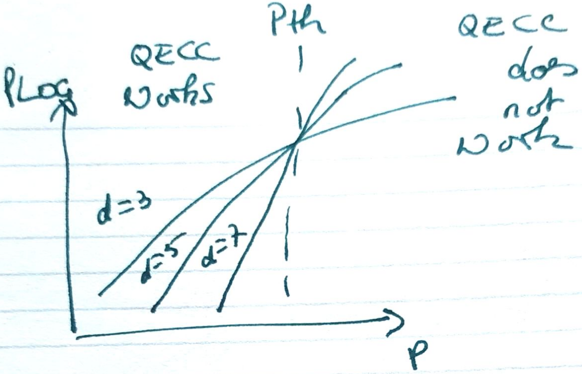
prob failure / Volume.



prob of a cube failing.

Noise Model : single qubit p
two qubit 10p
measurement 10p?

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



threshold prob. pth

$$P_{Log} \sim \left(\frac{P}{Pth}\right)^{\frac{d+1}{2}}$$

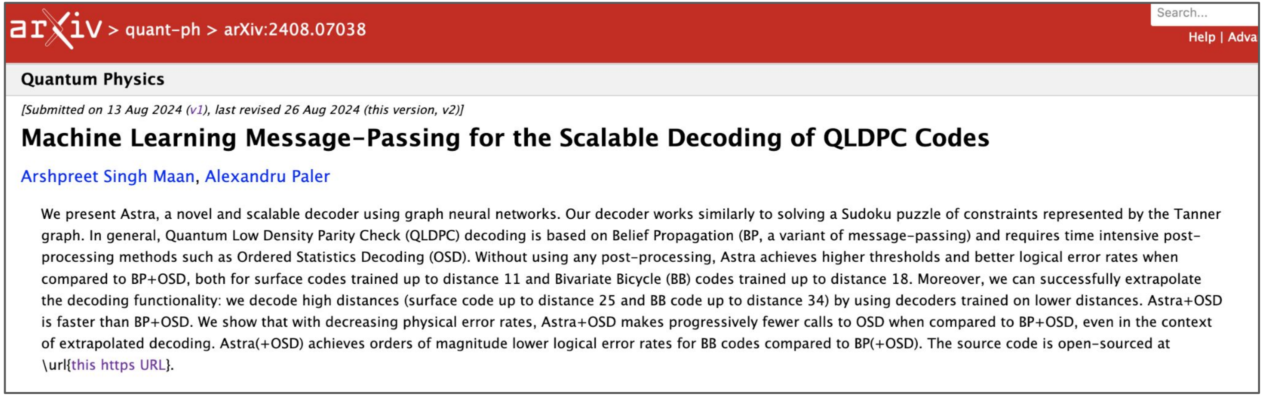
Usual values: p ~ 0.1%

pth ~ 1%

increase d by 2 → 10x lower P_{log}

d = 17 min. reg. for 10^9

Scalable (Machine Learning) Decoders



Quantum Physics

[Submitted on 13 Aug 2024 (v1), last revised 26 Aug 2024 (this version, v2)]

Machine Learning Message-Passing for the Scalable Decoding of QLDPC Codes

Arshpreet Singh Maan, Alexandru Paler

We present Astra, a novel and scalable decoder using graph neural networks. Our decoder works similarly to solving a Sudoku puzzle of constraints represented by the Tanner graph. In general, Quantum Low Density Parity Check (QLDPC) decoding is based on Belief Propagation (BP, a variant of message-passing) and requires time intensive post-processing methods such as Ordered Statistics Decoding (OSD). Without using any post-processing, Astra achieves higher thresholds and better logical error rates when compared to BP+OSD, both for surface codes trained up to distance 11 and Bivariate Bicycle (BB) codes trained up to distance 18. Moreover, we can successfully extrapolate the decoding functionality: we decode high distances (surface code up to distance 25 and BB code up to distance 34) by using decoders trained on lower distances. Astra+OSD is faster than BP+OSD. We show that with decreasing physical error rates, Astra+OSD makes progressively fewer calls to OSD when compared to BP+OSD, even in the context of extrapolated decoding. Astra(+OSD) achieves orders of magnitude lower logical error rates for BB codes compared to BP(+OSD). The source code is open-sourced at \url{this https URL}.

under consideration at PRX Quantum

ML Decoders: Introduction and Motivation

Optimal Decoding of QECC is a hard problem [1]

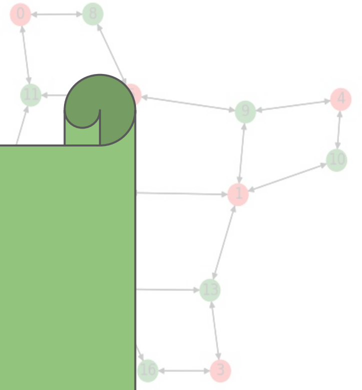
Belief propagation (BP) - one of the best-known classical decoding algorithms

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

Are



rface code of
tices are check
ces are data nodes

Neu etwork (NN) decoding has constant decoding runtime ☺

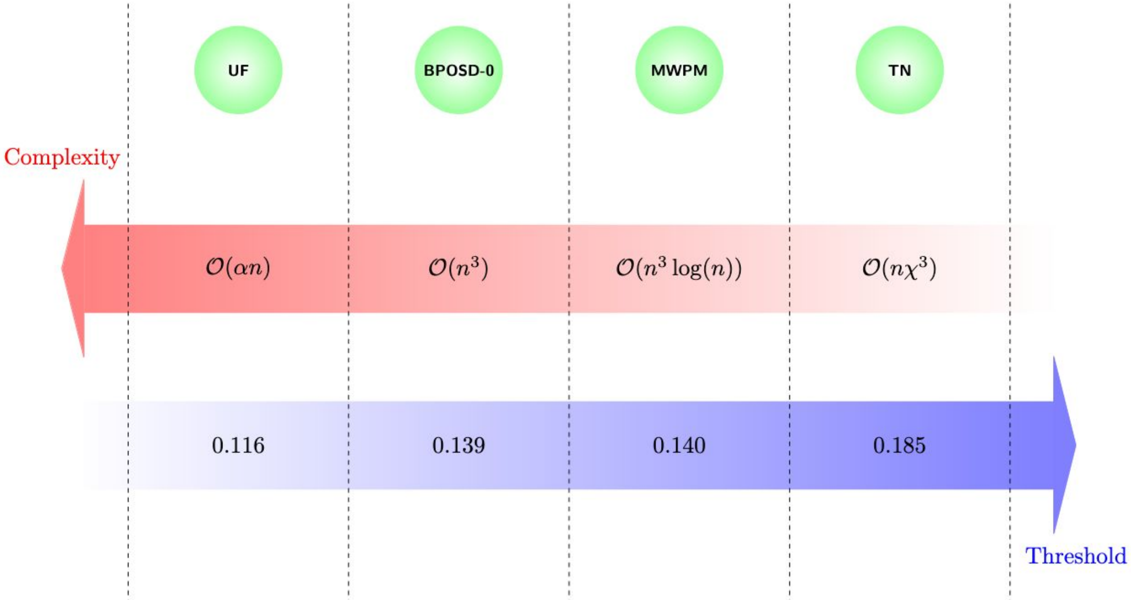
Limitations of previous NN based decoding approaches:

- 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

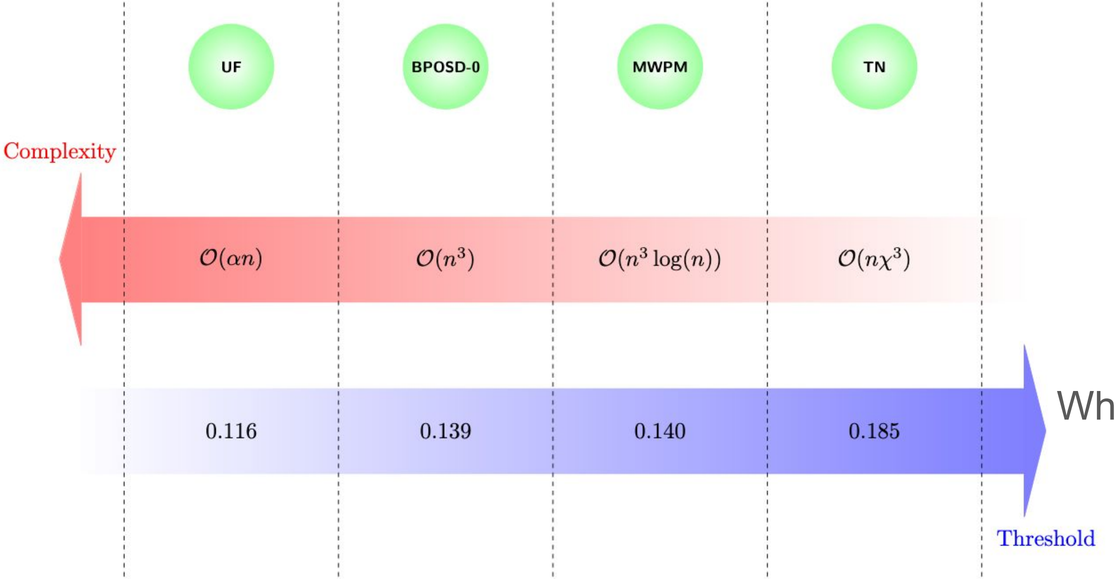
Why ML Decoders?

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?

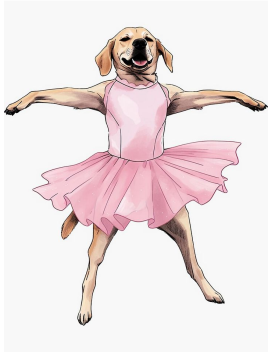


iOlius, A. D., Fuentes, P., Orús, R., Crespo, P. M., & Martinez, J. E. (2023). Decoding algorithms for surface codes. *arXiv preprint arXiv:2307.14989*.

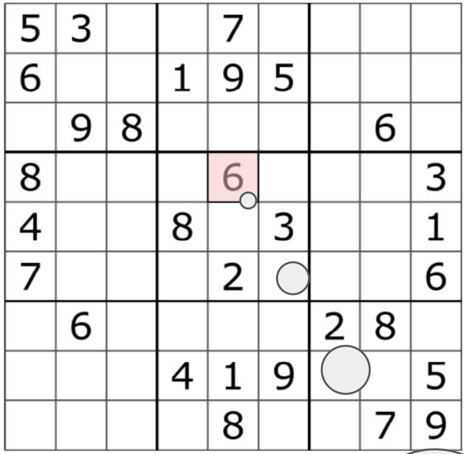
ML Decoding has linear time (although the scaling of the models with code distance is not known)

What the goal is:

What the state of the art is:

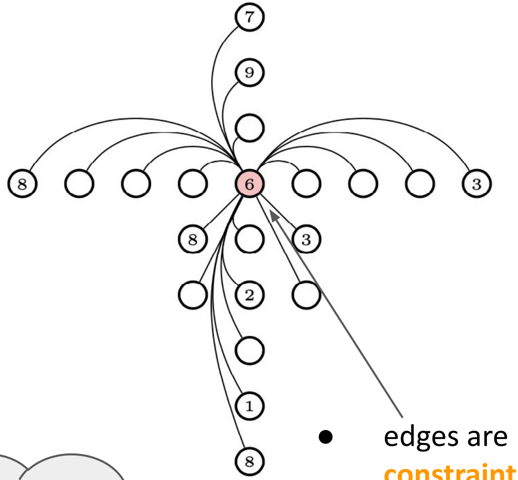


Astra: A Graph Neural Network (GNN) Decoder Learning BP to Satisfy Constraints

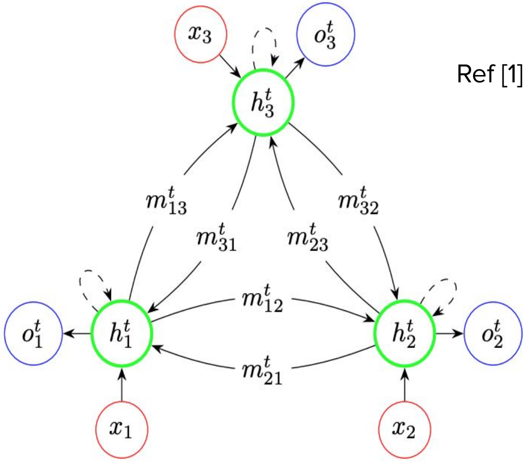


5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

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



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



Ref [1]

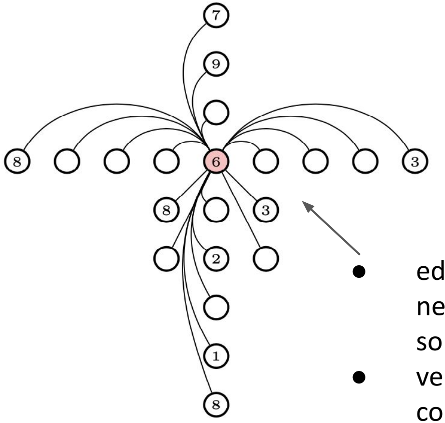
red: input vertices in GNN
blue: output
green: node state
messages are sent along the edges

[1] https://arxiv.org/abs/1711.08028

Astra: A Graph Neural Network (GNN) Decoder

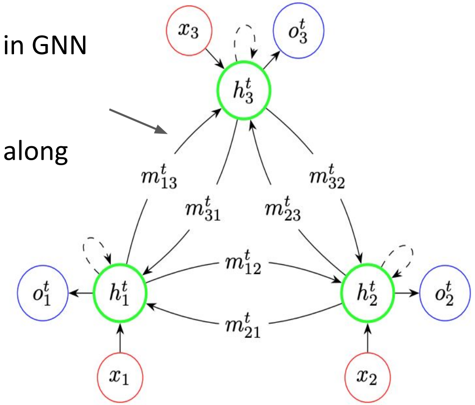
The Sudoku analogy - Learning BP

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



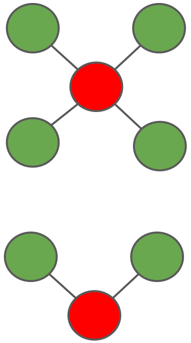
- edges are constraints necessary for the solution
- vertices are forming constraint pairs

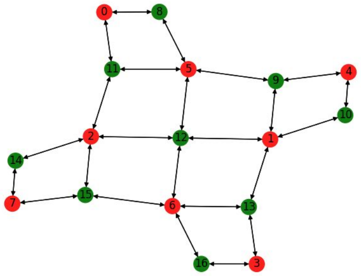
red: input vertices in GNN
blue: output
green: node state
messages are sent along the edges



?		?		?
	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

Astra as replacement of BP+OSD for Surface code

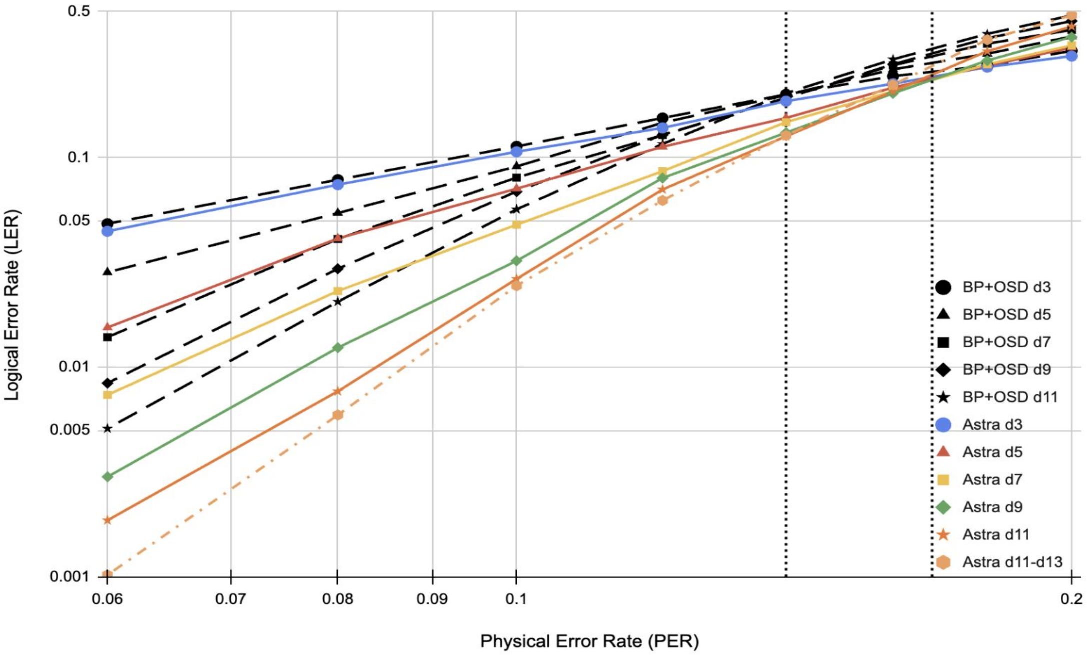


FIG. 1. The Logical Error Rate (LER) of Astra vs BP+OSD under code capacity depolarizing noise. Our decoder has a threshold of $\sim 17\%$, and BP+OSD has a threshold of $\sim 14\%$.

Extrapolated Astra+OSD vs BP+OSD for Surface code

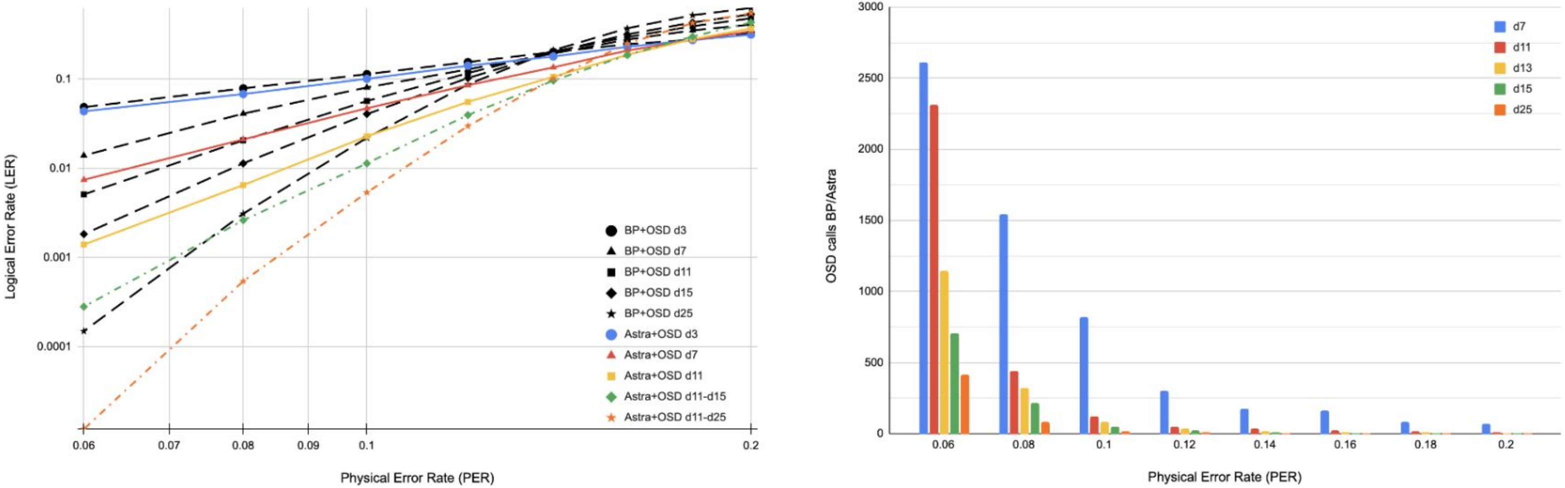


FIG. 4. Decoding surface codes with Astra+OSD vs BP+OSD by using OSD0 in the second stage. a) Astra+OSD achieves orders of magnitude better LER than BP+OSD and requires fewer OSD calls; b) Speedups of Astra+OSD vs BP+OSD are obtained because Astra converges more often than BP and, consequently, the OSD stage is called significantly fewer times. This holds even when performing extrapolated decoding with the d11 decoder e.g. for distance 25, at 0.06 error rate, Astra+OSD is 400x faster than BP+OSD.

Astra as replacement of BP+OSD for IBM’s BB code

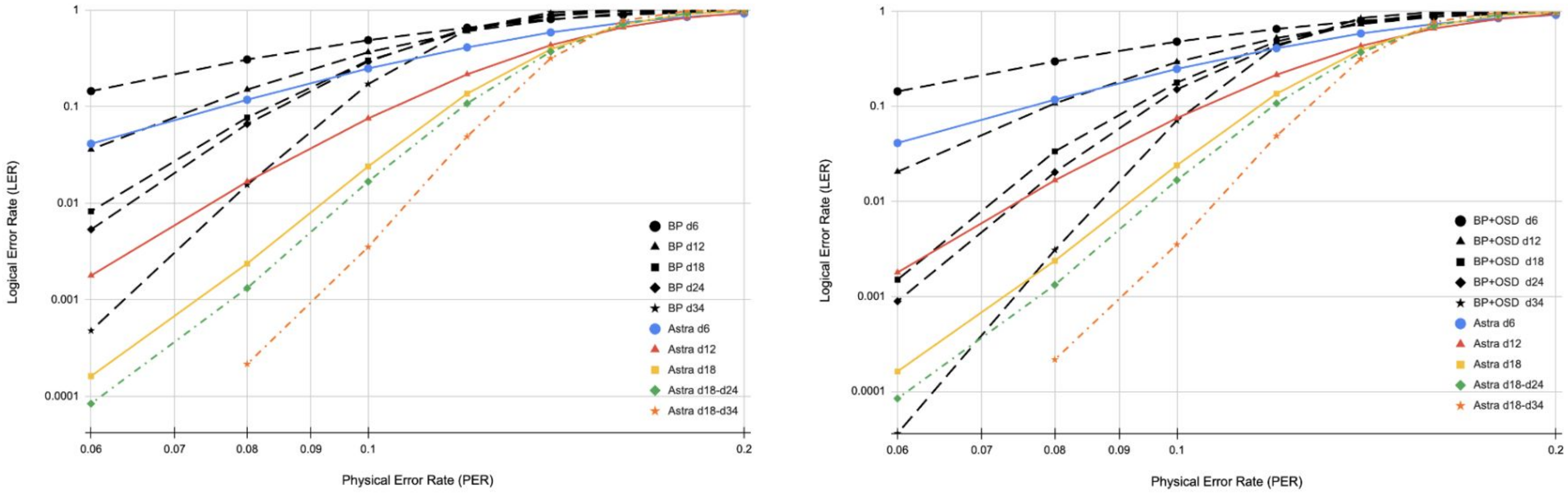


FIG. 5. Decoding BB codes with Astra compared to BP and BP+OSD. a) The LER of Astra is significantly lower compared to pure BP; b) The LER of Astra compared to BP+OSD.

Extrapolated Astra+OSD vs BP+OSD for IBM’s BB code

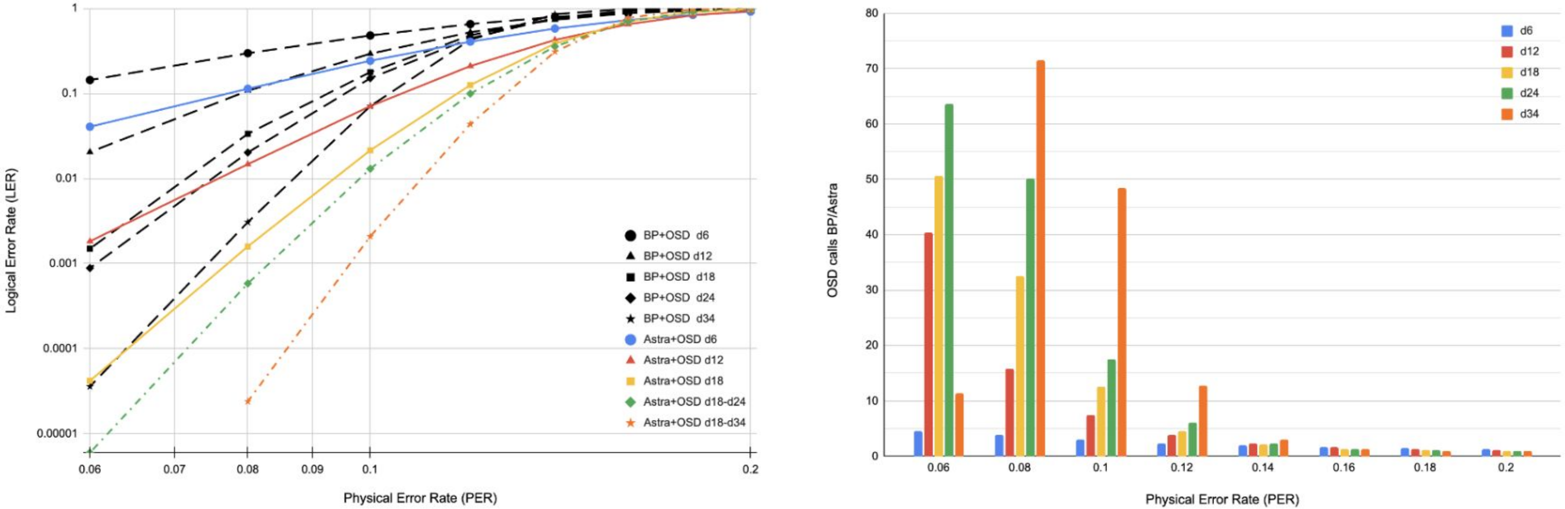
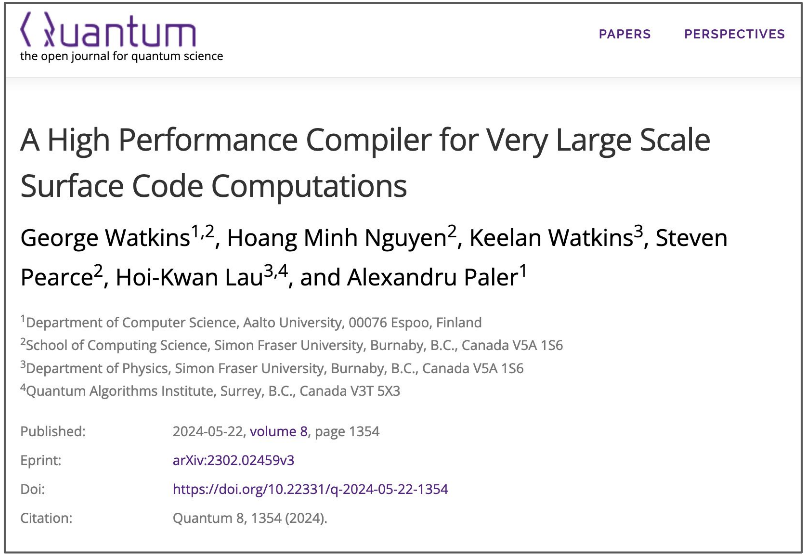


FIG. 6. Decoding BB codes with Astra+OSD compared to BP and BP+OSD. a) LER of Astra+OSD vs BP+OSD; b) Speed-up of Astra+OSD vs BP+OSD, Astra+OSD is $\sim 50x$ faster than BP+OSD for larger codes at low errors rates. The speedups persists even for the extrapolated decoding case of distance 24 and 34 using distance 18 GNN decoder.

Very Fast Compilers (for Lattice Surgery)



A High Performance Compiler for Very Large Scale Surface Code Computations

George Watkins[1,2], Hoang Minh Nguyen[2], Keelan Watkins[3], Steven Pearce[2], Hoi-Kwan Lau[3,4], and Alexandru Paler[1]

[1]Department of Computer Science, Aalto University, 00076 Espoo, Finland
[2]School of Computing Science, Simon Fraser University, Burnaby, B.C., Canada V5A 1S6
[3]Department of Physics, Simon Fraser University, Burnaby, B.C., Canada V5A 1S6
[4]Quantum Algorithms Institute, Surrey, B.C., Canada V3T 5X3

Published: 2024-05-22, volume 8, page 1354
Eprint: arXiv:2302.02459v3
Doi: https://doi.org/10.22331/q-2024-05-22-1354
Citation: Quantum 8, 1354 (2024).

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

Unitary Fund

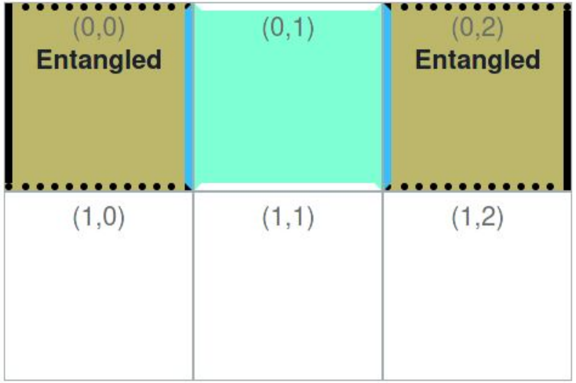
SFU SIMON FRASER UNIVERSITY

Aalto University

Our Challenge: *Logical Computations at scale*

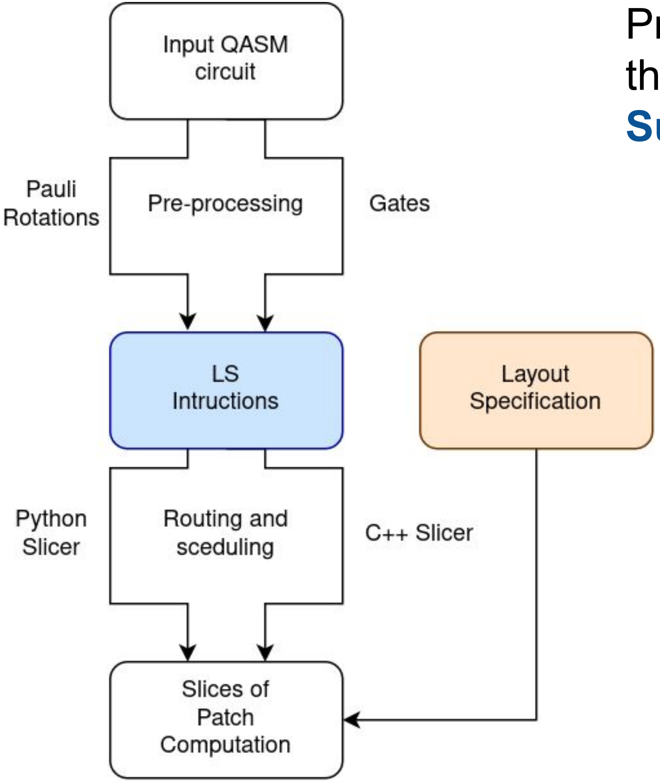
100s to 1000s of logical qubits

- 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

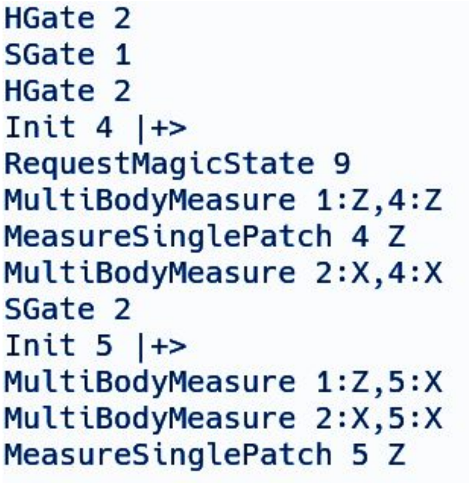


LS Compiler Architecture

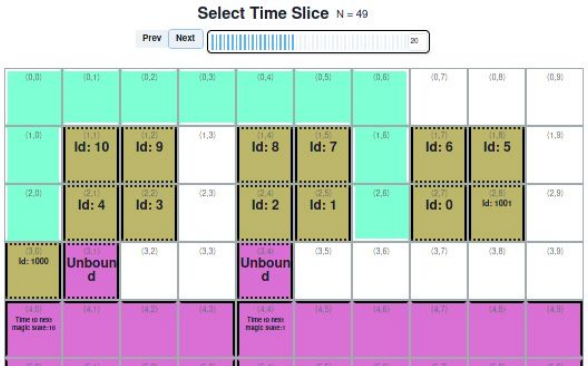
A pluggable pipeline in decoupled stages, with options and **text-based intermediate representations**



Pre-processing is decoupled from routing on the lattice thanks to an intermediate representation of **Lattice Surgery Instructions** and a **Layout Specification**

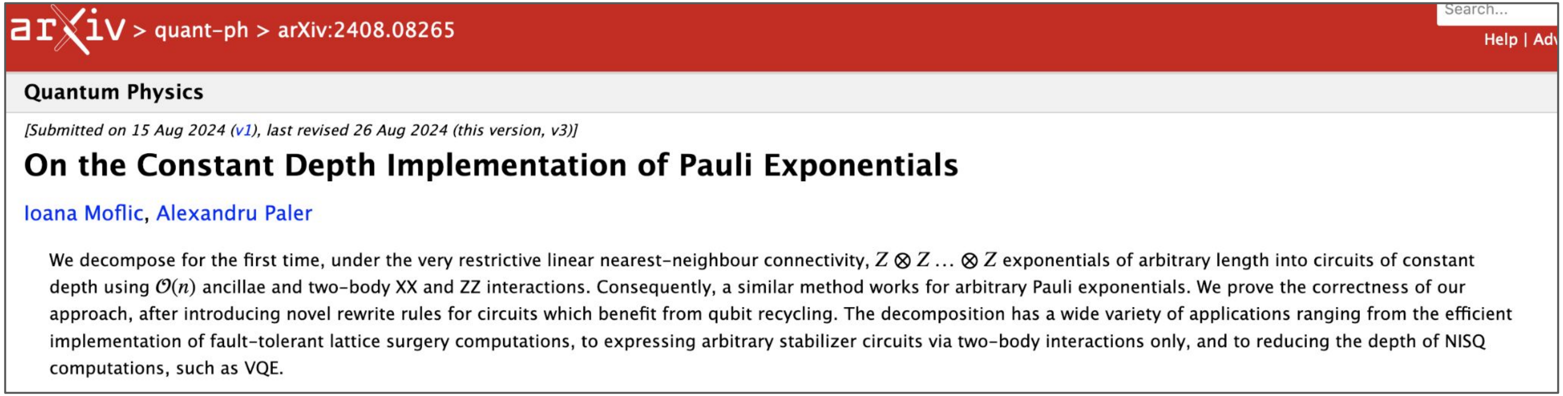


```
HGate 2
SGate 1
HGate 2
Init 4 |+>
RequestMagicState 9
MultiBodyMeasure 1:Z,4:Z
MeasureSinglePatch 4 Z
MultiBodyMeasure 2:X,4:X
SGate 2
Init 5 |+>
MultiBodyMeasure 1:Z,5:X
MultiBodyMeasure 2:X,5:X
MeasureSinglePatch 5 Z
```



⟨0,0⟩	⟨0,1⟩	⟨0,2⟩	⟨0,3⟩	⟨0,4⟩	⟨0,5⟩	⟨0,6⟩	⟨0,7⟩	⟨0,8⟩	⟨0,9⟩
⟨1,0⟩	⟨1,1⟩ Id: 10	⟨1,2⟩ Id: 9	⟨1,3⟩	⟨1,4⟩ Id: 8	⟨1,5⟩ Id: 7	⟨1,6⟩	⟨1,7⟩ Id: 6	⟨1,8⟩ Id: 5	⟨1,9⟩
⟨2,0⟩	⟨2,1⟩ Id: 4	⟨2,2⟩ Id: 3	⟨2,3⟩	⟨2,4⟩ Id: 2	⟨2,5⟩ Id: 1	⟨2,6⟩	⟨2,7⟩ Id: 0	⟨2,8⟩ Id: 1001	⟨2,9⟩
⟨3,0⟩ Id: 1000	⟨3,1⟩ Unbound	⟨3,2⟩	⟨3,3⟩	⟨3,4⟩ Unbound	⟨3,5⟩	⟨3,6⟩	⟨3,7⟩	⟨3,8⟩	⟨3,9⟩
⟨4,0⟩ Time to next magic state:10	⟨4,1⟩	⟨4,2⟩	⟨4,3⟩	⟨4,4⟩ Time to next magic state:1	⟨4,5⟩	⟨4,6⟩	⟨4,7⟩	⟨4,8⟩	⟨4,9⟩

Very Large Scale Circuit Optimizer



Quantum Physics

[Submitted on 15 Aug 2024 (v1), last revised 26 Aug 2024 (this version, v3)]

On the Constant Depth Implementation of Pauli Exponentials

Ioana Moflic, Alexandru Paler

We decompose for the first time, under the very restrictive linear nearest-neighbour connectivity, $Z \otimes Z \ldots \otimes Z$ exponentials of arbitrary length into circuits of constant depth using $\mathcal{O}(n)$ ancillae and two-body XX and ZZ interactions. Consequently, a similar method works for arbitrary Pauli exponentials. We prove the correctness of our approach, after introducing novel rewrite rules for circuits which benefit from qubit recycling. The decomposition has a wide variety of applications ranging from the efficient implementation of fault-tolerant lattice surgery computations, to expressing arbitrary stabilizer circuits via two-body interactions only, and to reducing the depth of NISQ computations, such as VQE.

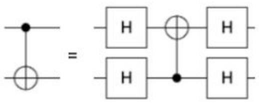
under consideration at PR Letters

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.



j) REVERSE

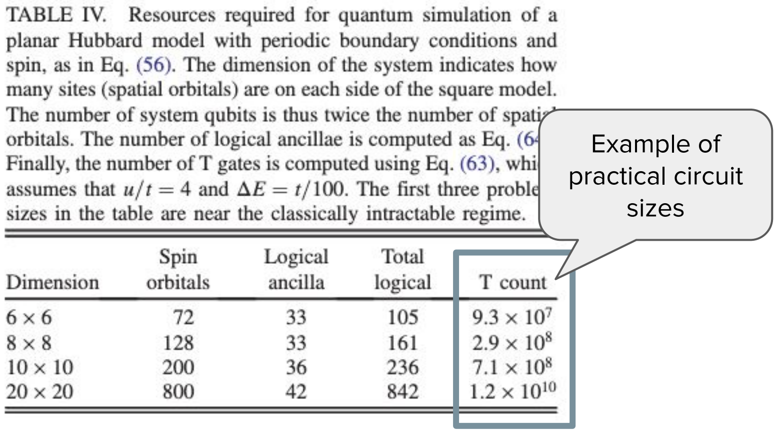


TABLE IV. Resources required for quantum simulation of a planar Hubbard model with periodic boundary conditions and spin, as in Eq. (56). The dimension of the system indicates how many sites (spatial orbitals) are on each side of the square model. The number of system qubits is thus twice the number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

Dimension	Spin orbitals	Logical ancilla	Total logical	T count
6×6	72	33	105	9.3×10^7
8×8	128	33	161	2.9×10^8
10×10	200	36	236	7.1×10^8
20×20	800	42	842	1.2×10^{10}

Example of practical circuit sizes

Encoding Electronic Spectra in Quantum Circuits with Linear T Complexity

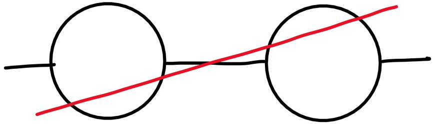
Ryan Babbush, Craig Gidney, Dominic W. Berry, Nathan Wiebe, Jarrod McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven
Phys. Rev. X **8**, 041015 – Published 23 October 2018

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

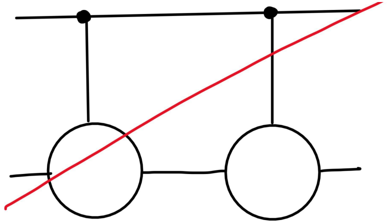
Methods

We consider four types of gate templates:

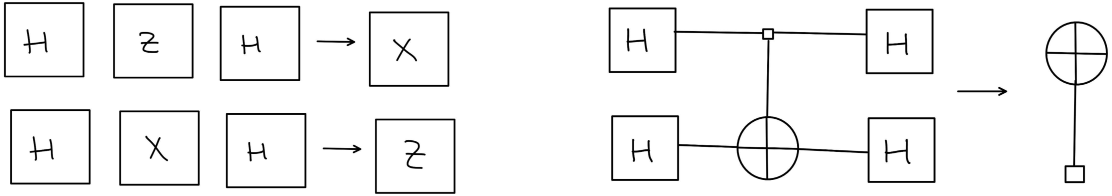
- **Single-qubit gate cancellations**



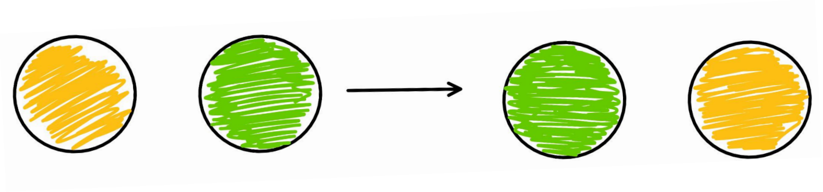
- **Two-qubit gate cancellations**



- **Base changes**



- **Commutations**

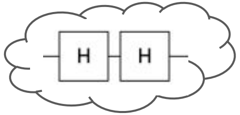


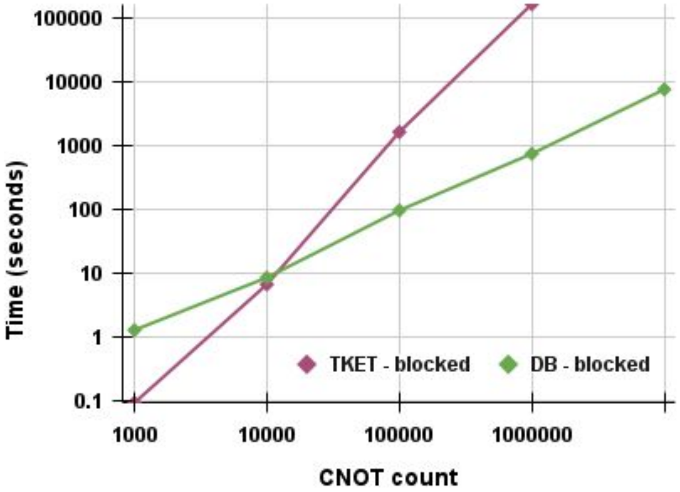
Results: Random Synthetic Circuits

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)



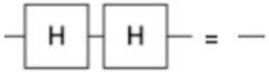


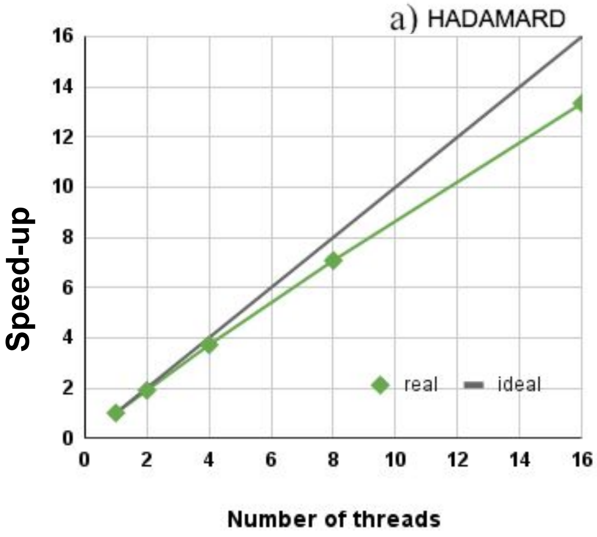
considered the fastest optimizer (written in Rust)

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

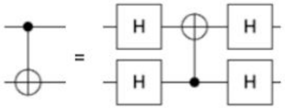
Results: Multi-threaded performance

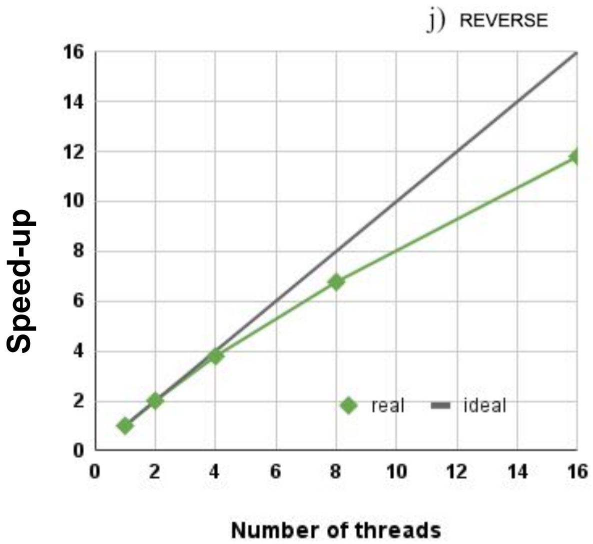
Type-1





Type-2





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 are learning the messages and algorithms of a message passing https://arxiv.org/pdf/2408.07038
2. **Large scale compilation and optimization**
   a. Engineering Reward Functions seems to speed/improve RL https://arxiv.org/abs/2311.12498
   b. Compression of RL states with autoencoders https://arxiv.org/abs/2303.03280
   c. Some tricks can massively improve the compilation https://arxiv.org/abs/2408.08265

funded by: DARPA QUANTERA ERA-NET Cofund in Quantum Technologies Google MINISTRY OF EDUCATION AND CULTURE

Aalto University