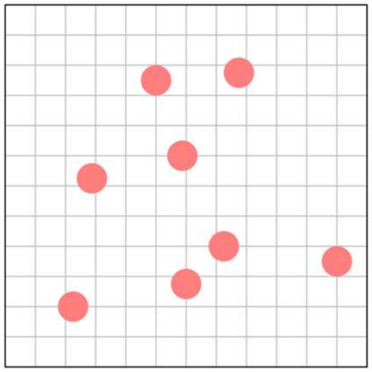
Quasilinear Time Decoding Algorithm for Topological Codes with High Error Threshold

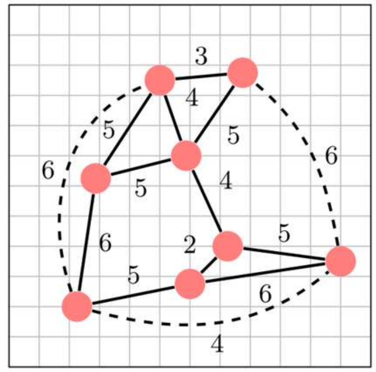
Mark Shui Hu B.A. Applied Physics
David Elkouss Prof.

Contents

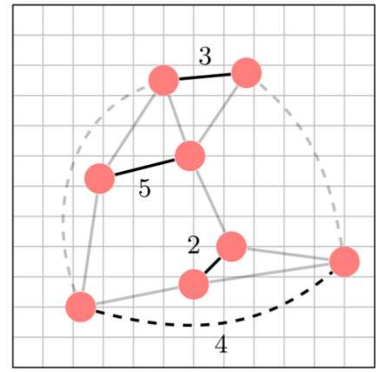
Minimum-Weight Perfect Matching decoder



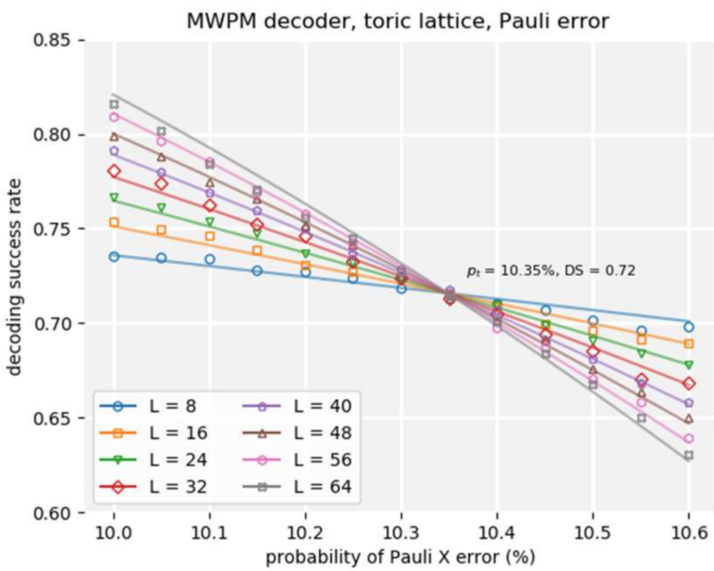
Measured syndrome

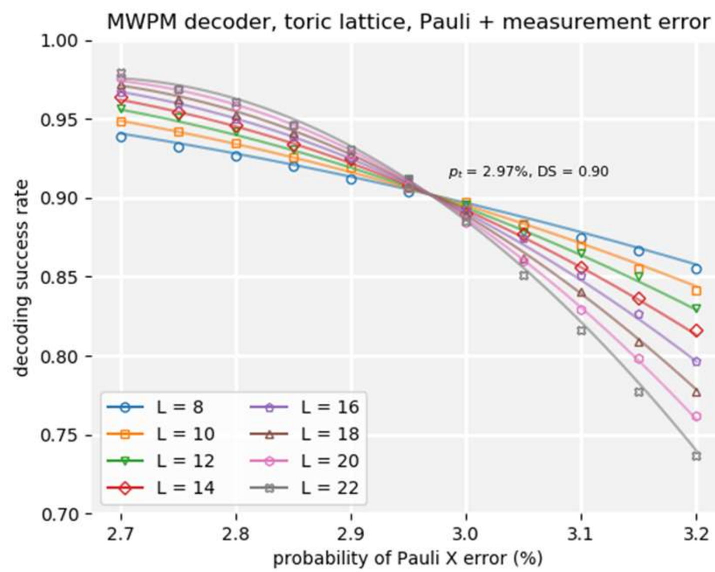


Completely connected graph with weights



Minimum weight subset of edges

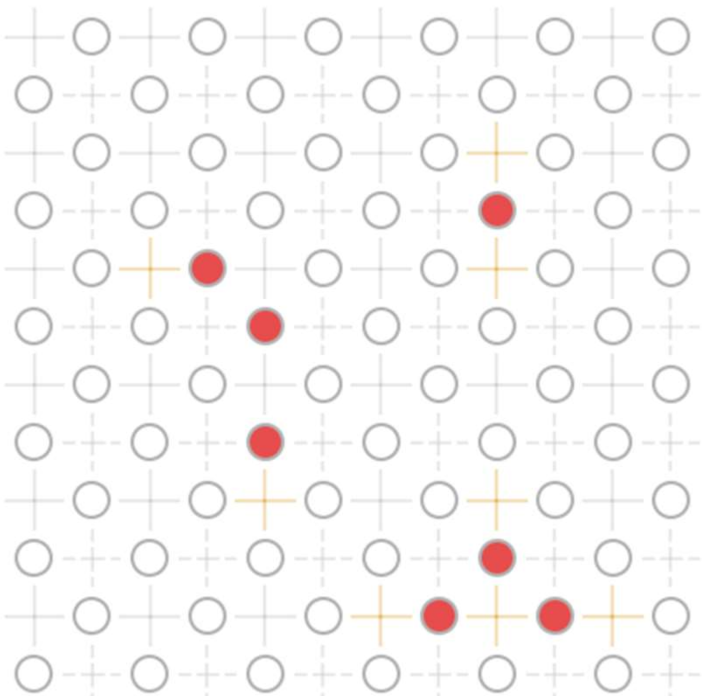




Worst case time complexity $\mathcal{O}(n^2\sqrt{n})$

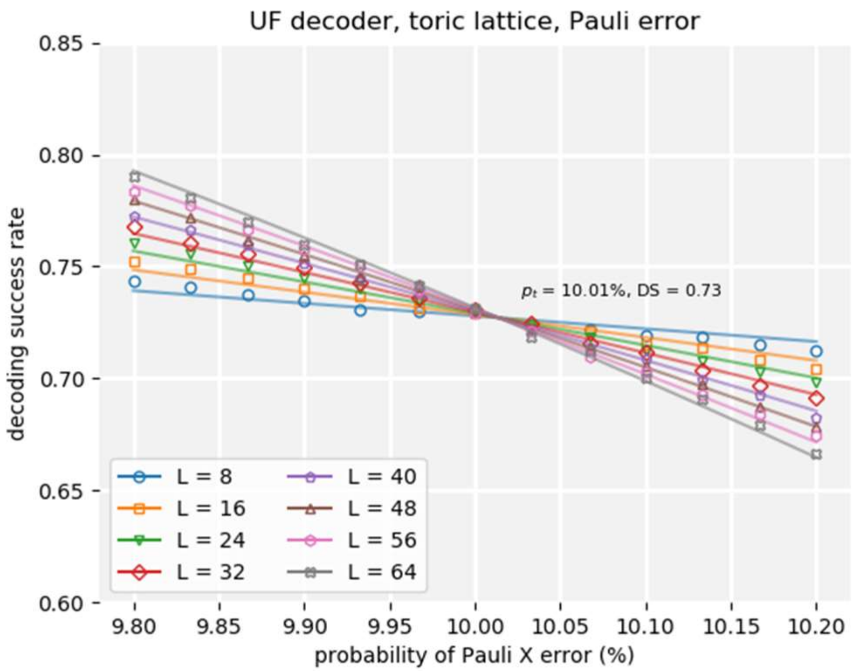
Union-Find decoder

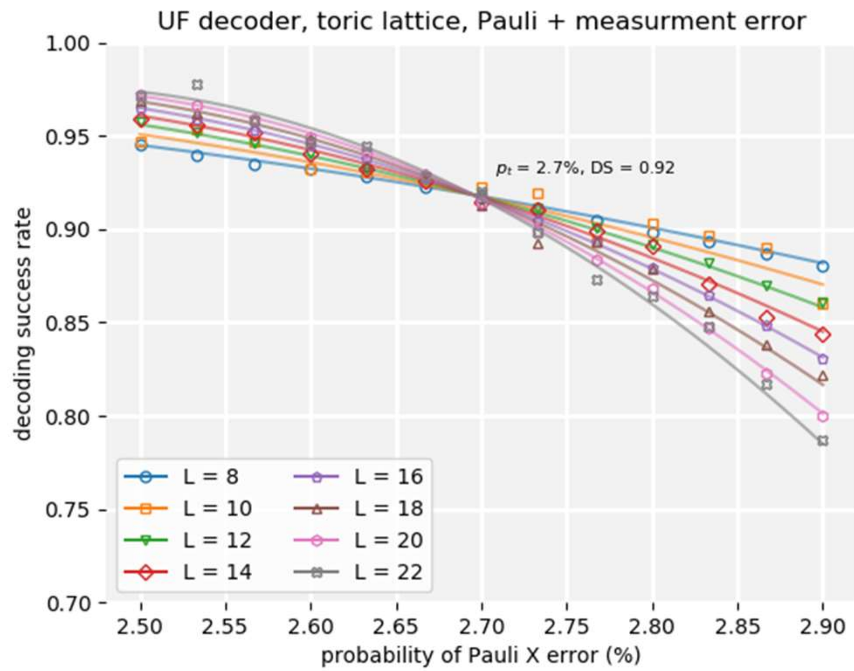
- Nontrivial syndrome: odd-parity *cluster*
- Grow odd-parity clusters in size until merged with other odd-parity cluster
- Apply weighted growth: order cluster growth by size
- Tree by DFS
- Peel cluster-trees



arXiv:1709.06218 Delfosse, N. and Nickerson N. H., Almost-linear time decoding algorithm for topological codes

Union-Find decoder performance



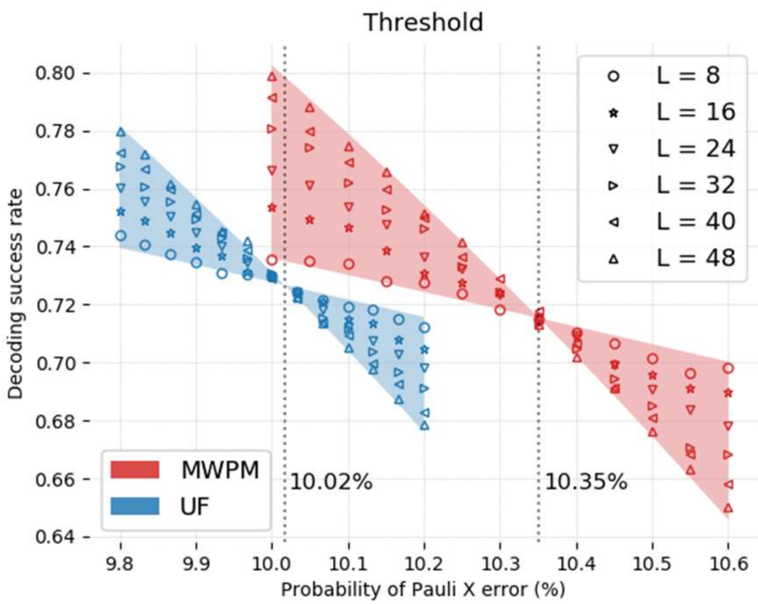


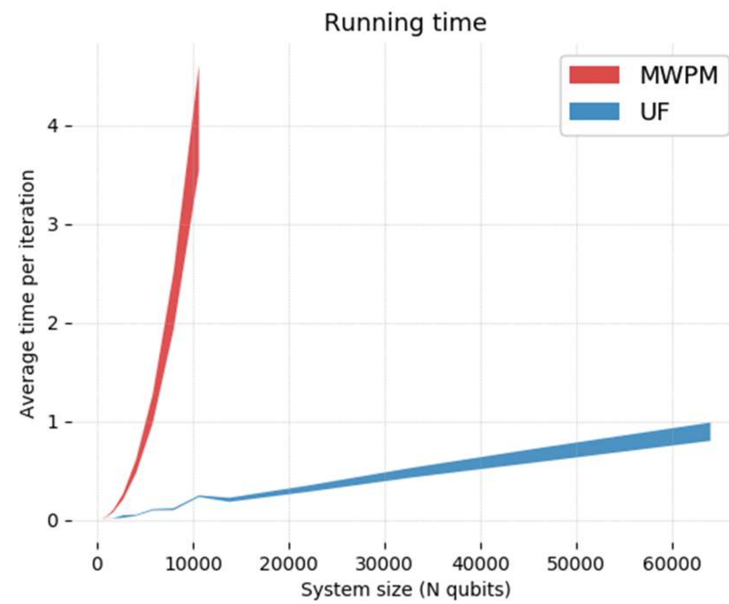
- Reported thresholds
  - toric 2D: 9.2% (weighted 9.9%)
  - toric 3D: 2.4% (weighted 2.6%)

arXiv:1709.06218 Delfosse, N. and Nickerson N. H., Almost-linear time decoding algorithm for topological codes

Decoders

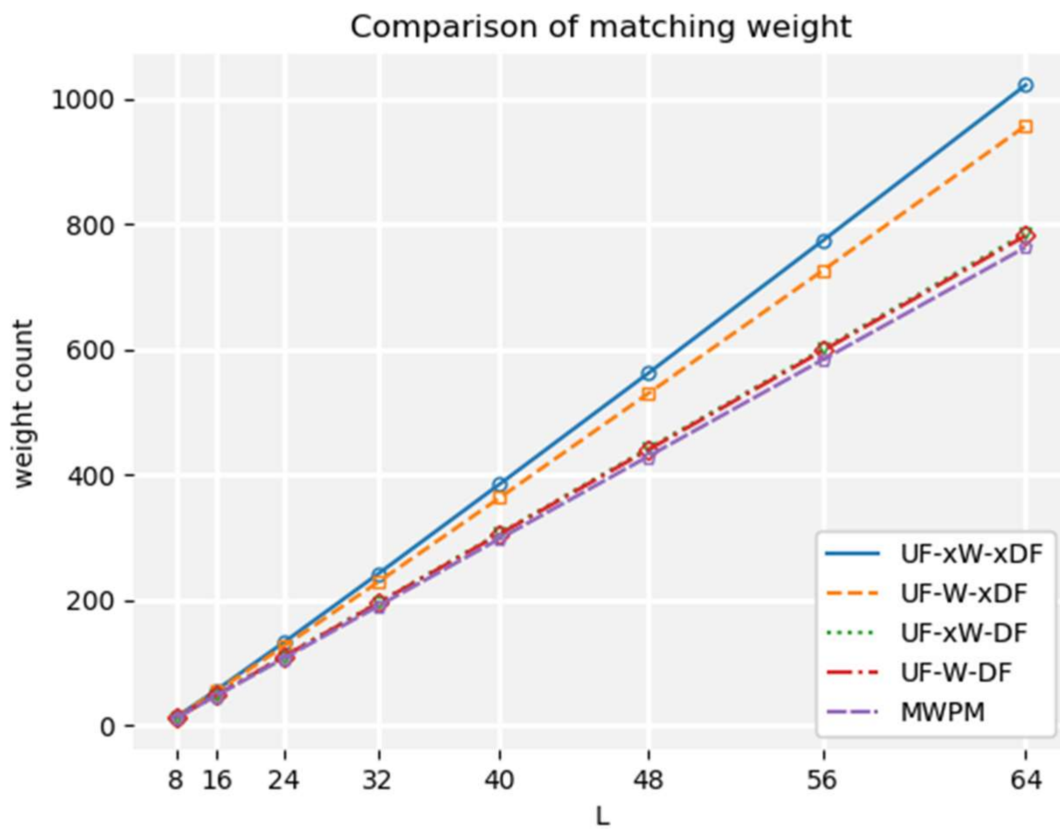
Decoder	2D toric threshold	3D toric threshold	Complexity
MWPM decoder	10.3% 10.35%	2.9% 2.97%	$\mathcal{O}(n^2\sqrt{n})$
UF decoder	9.9% 10.01%	2.6% 2.70%	$\mathcal{O}(n\alpha(n))$





Matching weight heuristic

Decoder	Threshold
UF-xW-xDF	9.71%
UF-xW-DF	9.79%
UF-W-xDF	9.98%
UF-W-DF	10.01%
MWPM	10.35%



Intuition that lower matching weight is a heuristic for increased threshold

Dynamic forests during growth

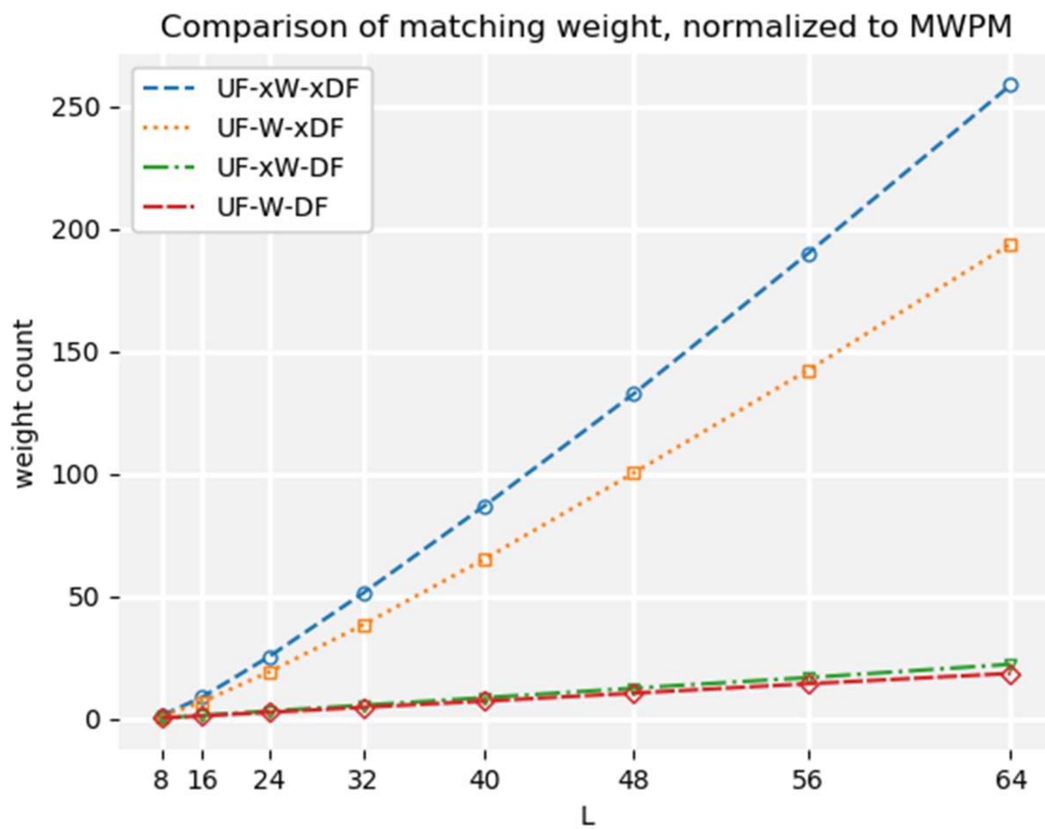
1. Grow cluster graph
2. Make cluster tree
3. Peel tree

1. Grow cluster tree
2. Peel tree

Dynamic forest equivalent to parallel breadth-first searches from the syndromes

Matching weight heuristic

Decoder	Threshold
UF-xW-xDF	9.71%
UF-xW-DF	9.79%
UF-W-xDF	9.98%
UF-W-DF	10.01%
MWPM	10.35%



Intuition that lower matching weight is a heuristic for increased threshold

Decoders

Intuition that lower matching weight is a heuristic for increased threshold

Decoder	2D toric threshold	3D toric threshold	Complexity
MWPM decoder	10.3% 10.35%	2.9% 2.97%	$\mathcal{O}(n^2\sqrt{n})$
UF decoder	9.9% 10.01%	2.6% 2.70%	$\mathcal{O}\big(n\alpha(n)\big)$

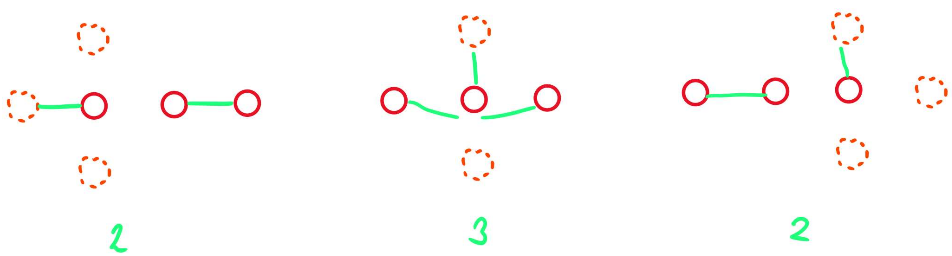
Union-Find Balanced-Bloom decoder

- Potential matching weight
- Data structure: node trees
- Node delays
- Calculation
- Performance

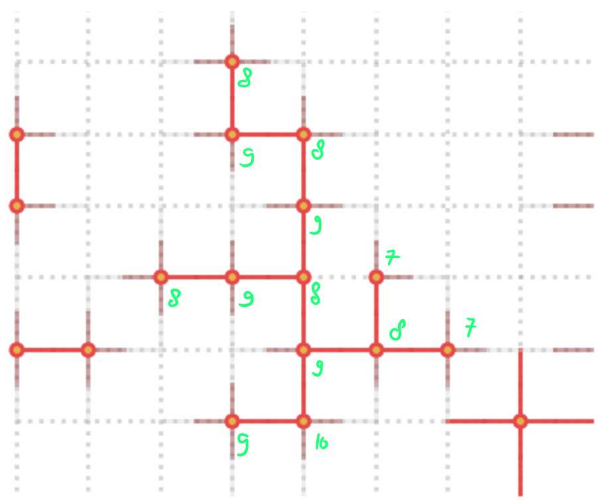
Potential matching weight

- Cluster C is a graph $\mathrm{C}(\mathcal{E}, \mathcal{V})$

- PMW$(v \in \mathcal{V}, C)$
  The matching weight of edges in E after a hypothetical union of C with another cluster on an edge supported by v.

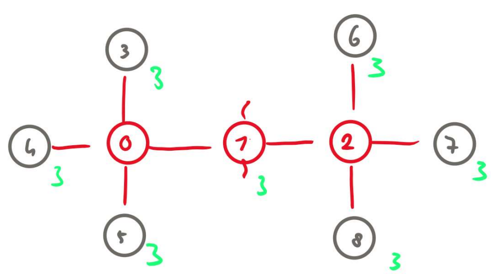


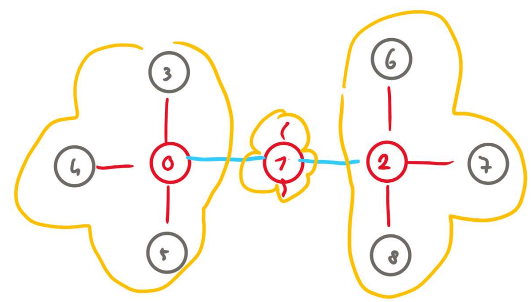
- <u>Minimize weight by prioritizing growth</u>
- Vertex specific
- *Local* variable



Node tree data structure

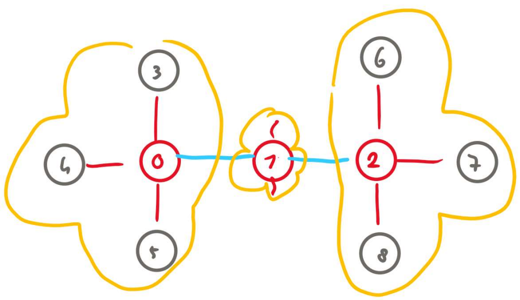
- node n represents a subset of vertices of a cluster C $\mathcal{V}_n \subseteq \mathcal{V}$ for which the vertices are *seeded* in the same vertex v_{seed}.
- Let a cluster by additionally represented by a set of nodes $\mathcal{N} = \{n_1, n_2, \dots\}$.





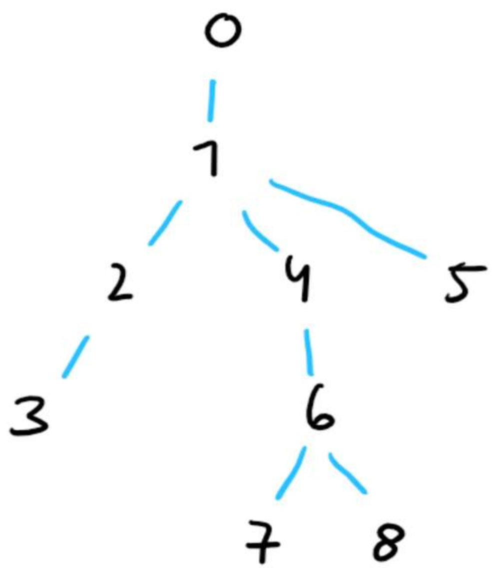
- Boundary vertices in the same node have equal PMW.
- Calculation of PMW reduced to the node tree
- Prioritize node specific growth with low PMW

Node delays



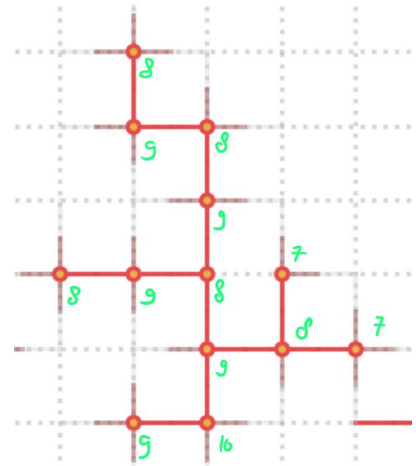
Bloom of a node:
Growing the boundaries belonging to a node n

Delayed bloom:
To suspend the bloom of a node for # iterations until an *equilibrium* of PMW is reached in the cluster
→ **Balanced Bloom**

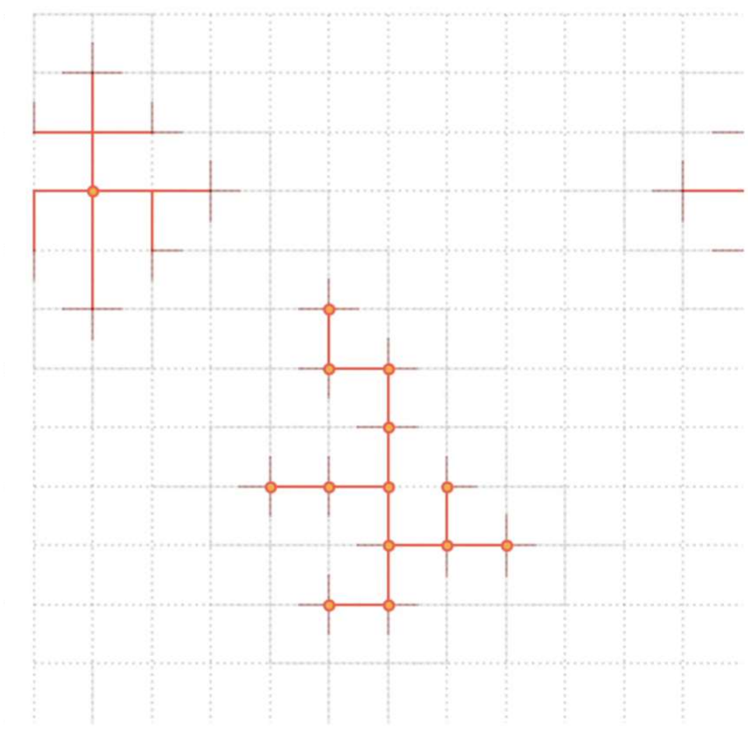


- Delay is calculated via DFS of the node tree
- $Delay(n_i) = f(n_i, n_{i-1})$

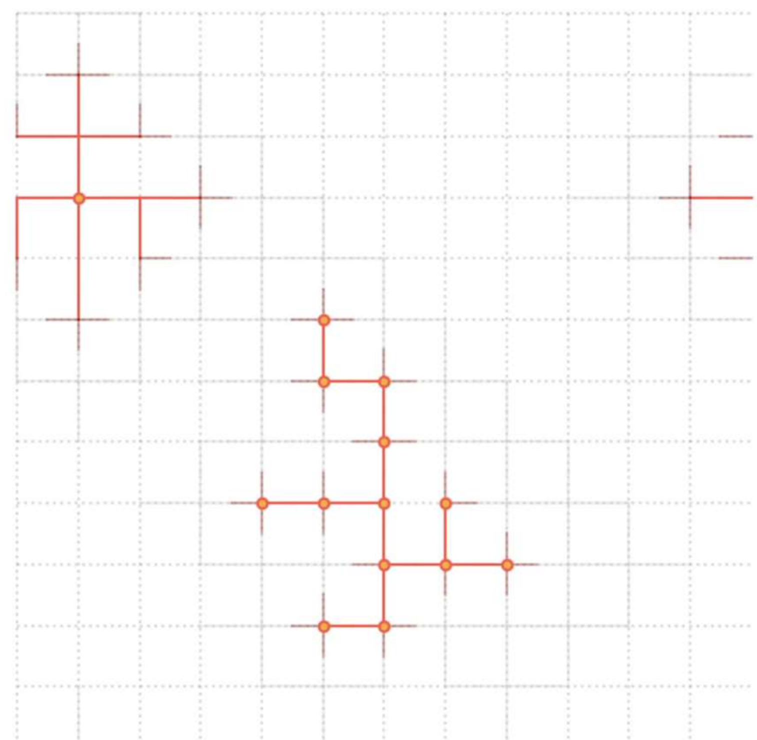
Comparison



Weighed Union-Find

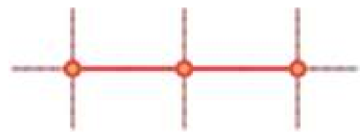


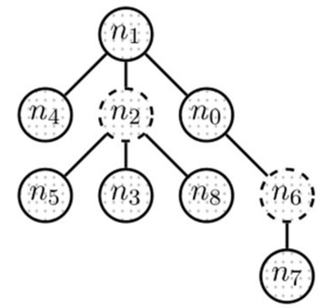
Union-Find Balanced-Bloom



Node delay calculation

$n.p$ parity





$n.d$ delay difference in PMW compared with root

$n.r$ radius number of blooms

$n.g$ $n.r$ mod 2

$$n_\beta.p = \begin{cases} 0, & \textit{if } n_\beta \textit{ has no children} \\ \left(\sum_j 1 - n_{\gamma,j}.p\right) \bmod 2 \mid n_\gamma \textit{ child of } n_\beta, & n_\beta \equiv s_\beta \\ 1 - \left(\sum_j 1 - n_{\gamma,j}.p\right) \bmod 2 \mid n_\gamma \textit{ child of } n_\beta, & n_\beta \equiv l_\beta \end{cases}$$

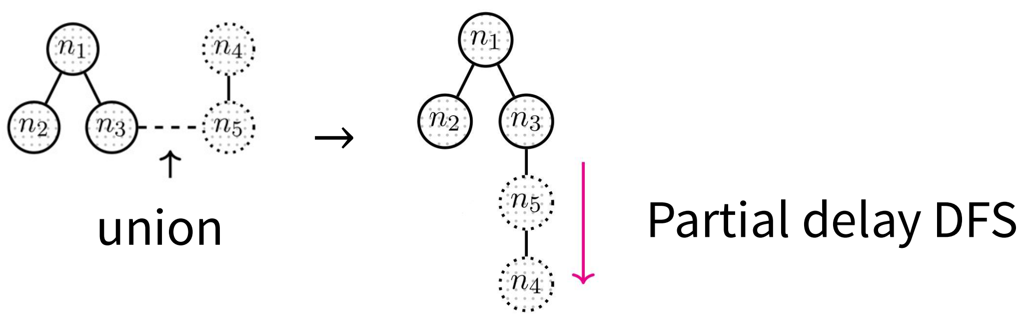
$$n_\beta.d = n_\alpha.d + \left[\left(\left\lfloor \frac{(n_\beta.r + n_\beta.g)}{2} \right\rfloor - \left\lfloor \frac{(n_\alpha.r + n_\beta.g)}{2} \right\rfloor + (-1)^{n_\beta.p+1}(n_\alpha, n_\beta) \right) - (n_\beta.g + n_\alpha.g) \bmod 2 \right] \mid n_r.d = 0, \; n_\beta \text{ child of } n_\alpha$$

Node tree unions

After union, tree structure changes: recalculate delay

- Union of odd/odd clusters → even cluster, no PMW
- Union of even/even clusters → even cluster, no PMW
- Union of odd/even clusters → odd cluster

Delay preservation union:
Preserve calculated delays in odd cluster by pointing merging node of the even cluster to the odd cluster.

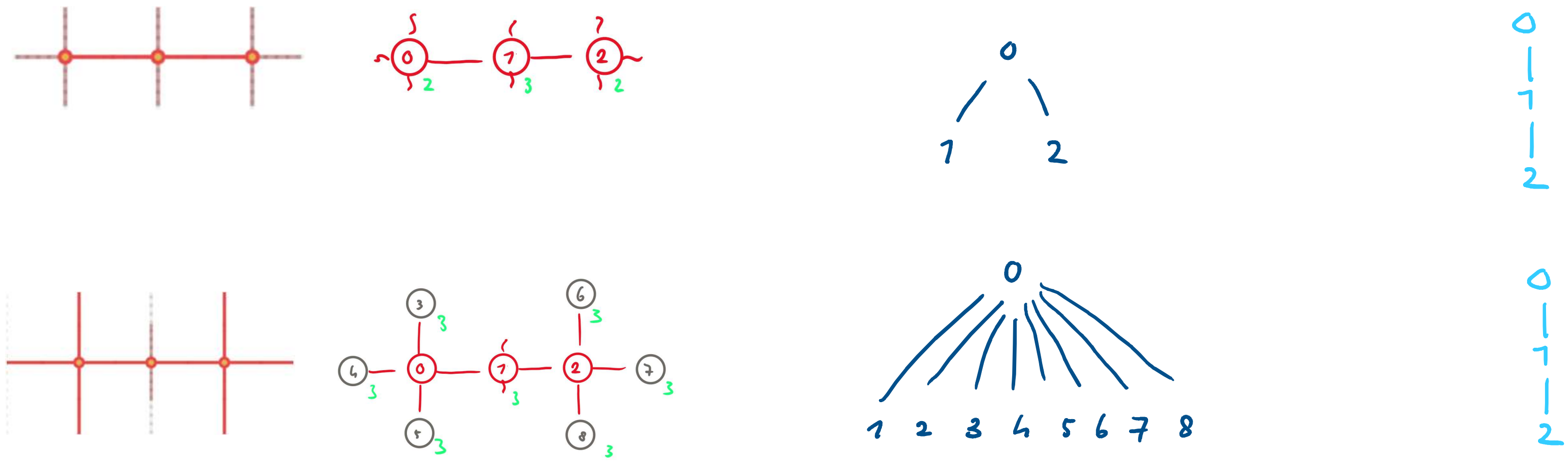


Worst case complexity $\mathcal{O}(n \log n)$

Relevant data structures

Union-Find
Vertex tree $\mathcal{V}$

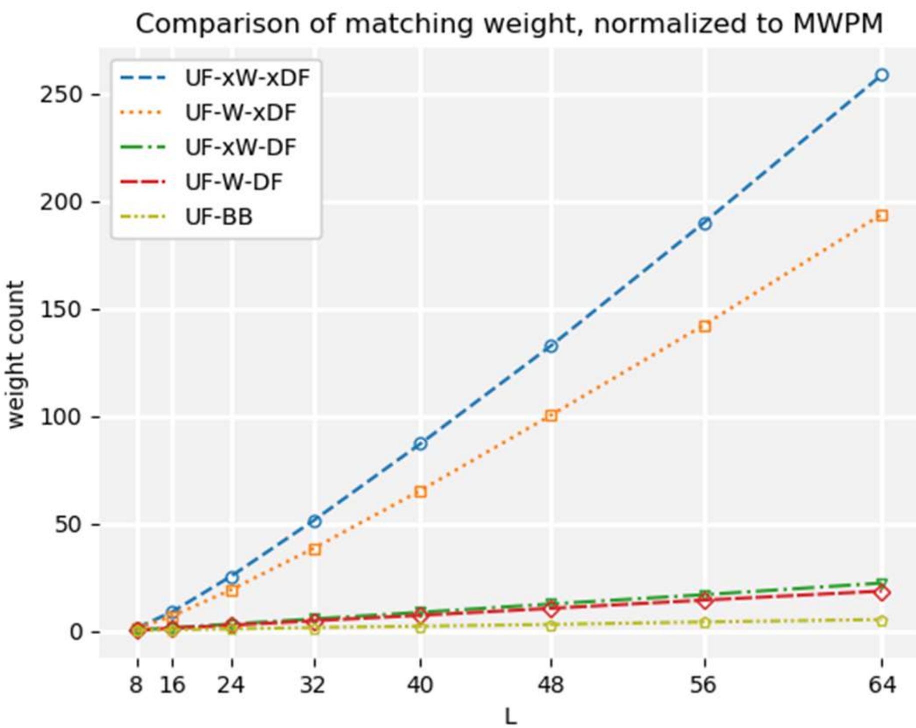
Node tree $\mathcal{N}$

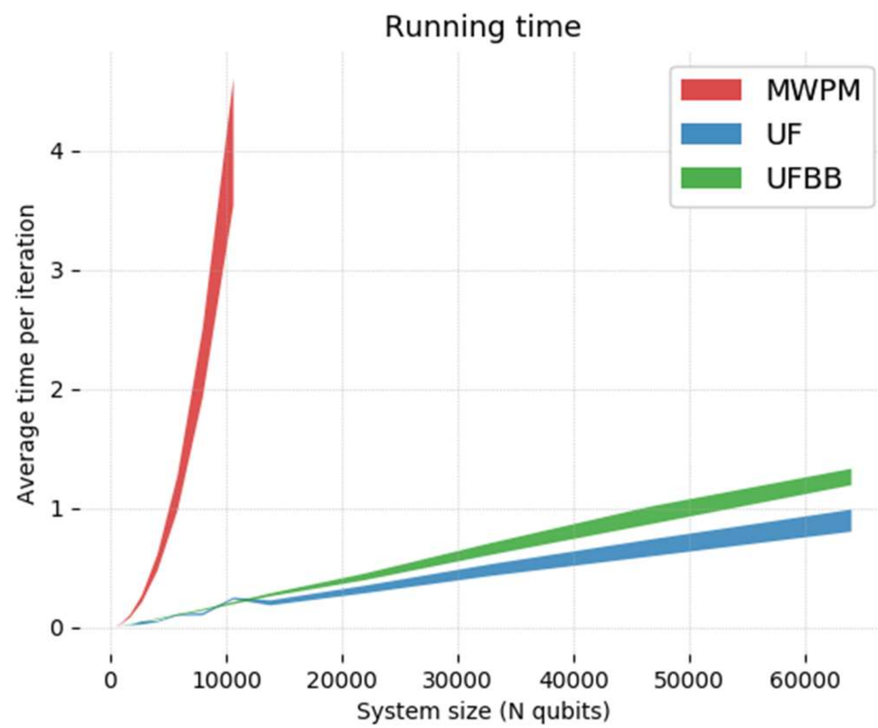


Function	Cluster identification	Potential matching weight
Tree structure	Edges are parent pointers	Reduced graph of cluster
Tree deformation	Path compression	No
Tree merge	Weighted union	Node tree union

Worst case complexity $\mathcal{O}(n \log n)$

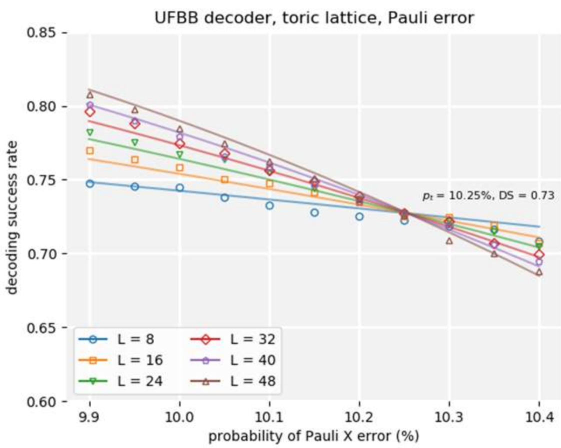
UF-BB performance

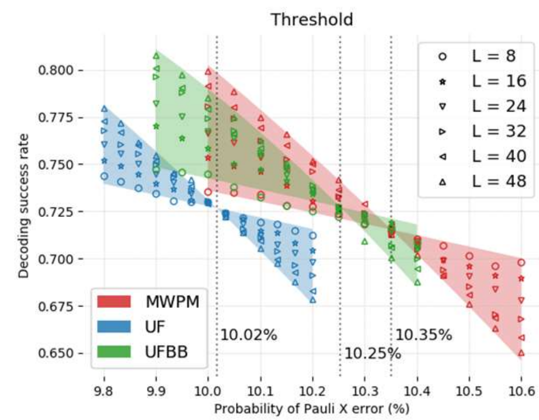


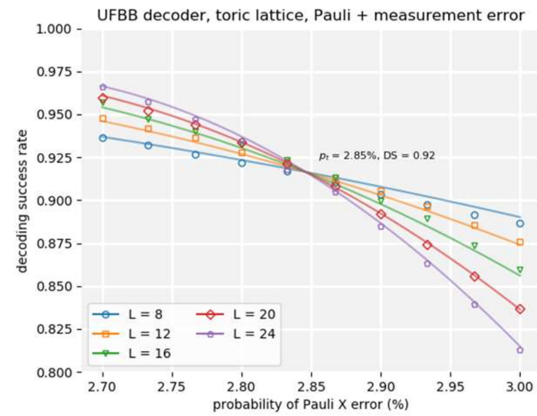


UF-BB threshold

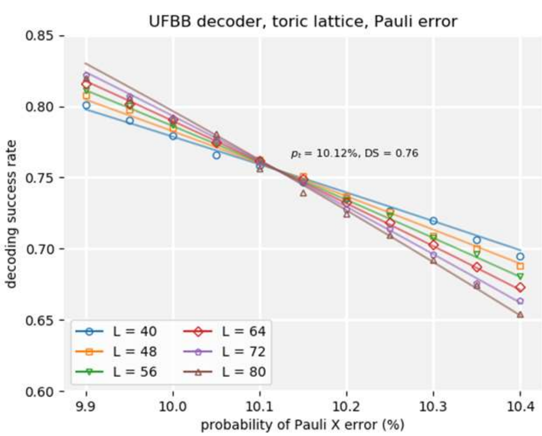
Investigate *small* lattice sizes

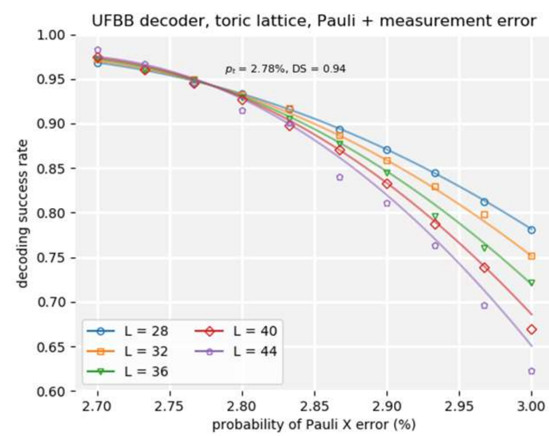




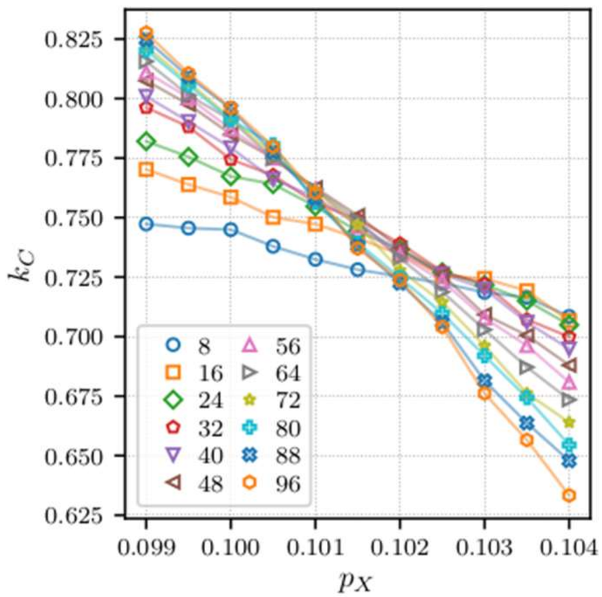


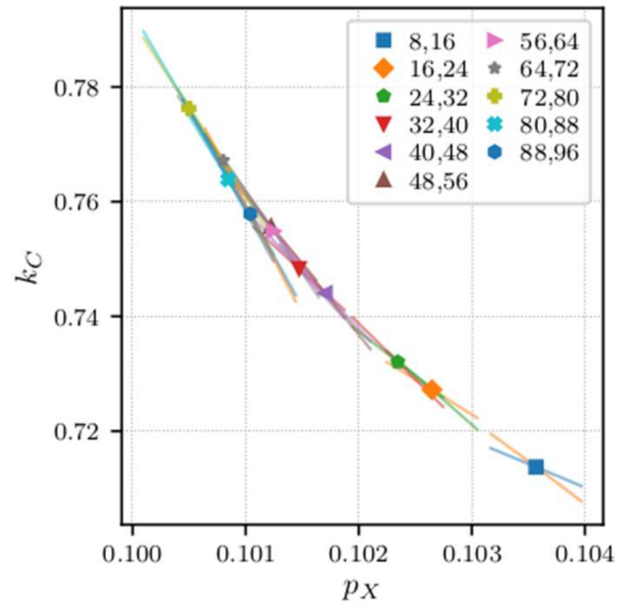
Investigate *bigger* lattice sizes

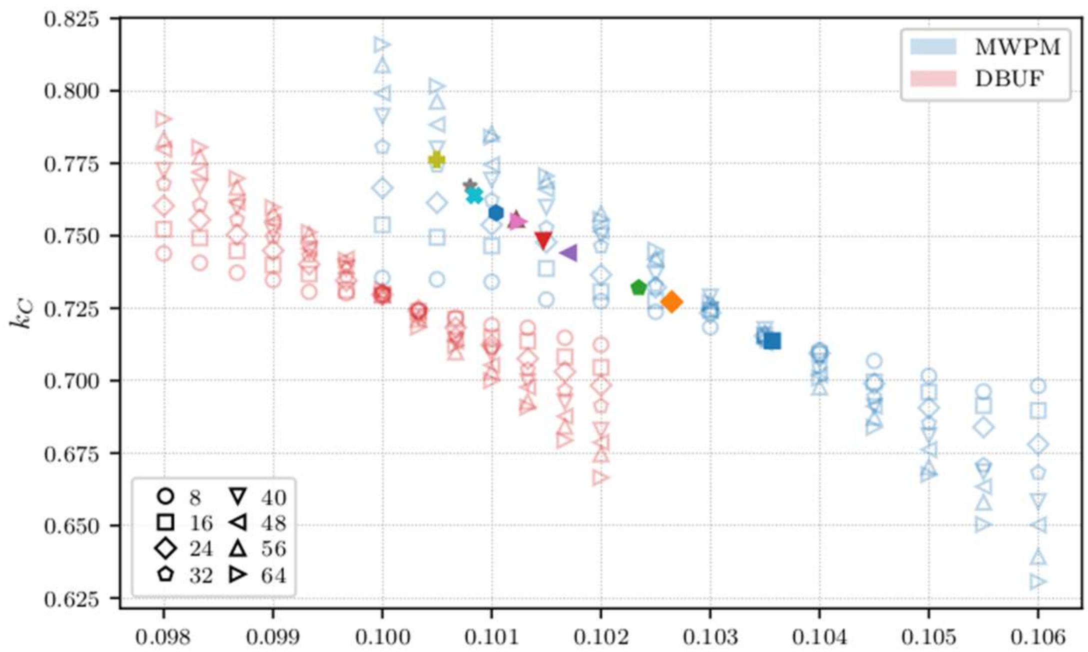




UF-BB threshold







Conclusions

- The performance of the UF-BB decoder is better than vanilla UF decoder for all lattice sizes and comparable to MWPM for small and medium lattice sizes with quasilinear complexity

- Lower matching weight is a heuristic for increased threshold

Code used for simulation, visualization can be found at
https://github.com/watermarkhu/oop_surface_code