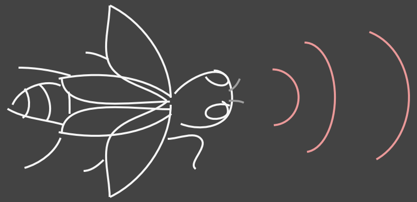
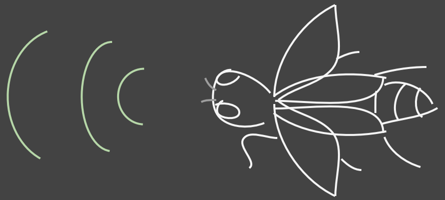



Quantum Routing

●●●

Aniruddha Bapat

with Eddie Schoute, Andrew Childs, Alexey Gorshkov,
REU-CAAR-20: Sam King and Hrishee Shastri
REU-CAAR-21: Sam DeCoster, Nicole Dong, Mason Wittman

Outline

- The routing problem
- REU-CAAR-20 showcase: Routing with fast reversal
- REU-CAAR-21: Routing with defects

Part I: Routing

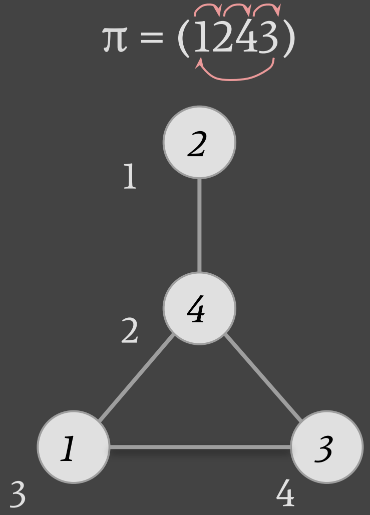
(Classical) Routing

Routing models the problem of information transfer in a network of connected information sources (such as CPUs in a cluster).

Setup: Given a graph *$G = (V, E)$*. At *$t=0$*, every node *i* has a “packet” with a vertex label *$\pi(i)$* on it, where *π* is a permutation of the nodes.

Allowed moves: swap packets between any two vertices connected by an edge. Each swap consumes 1 time step.

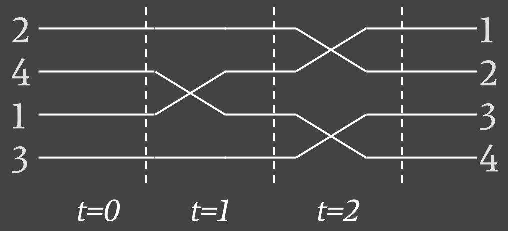
Goal: Route every packet to its given destination in the least possible time.



Example

$\pi = (1243) = (23)(12)(34).$

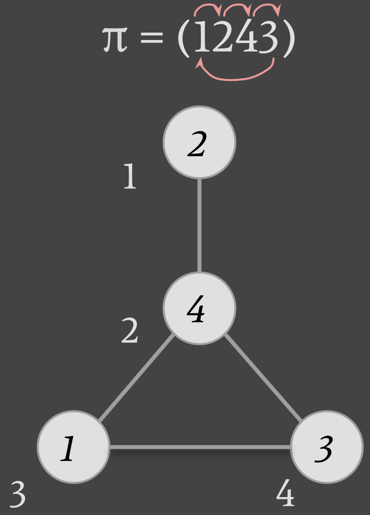
$C(\pi)$:



$\mathrm{rt}(G, \pi) := \min_{C(\pi)} \mathrm{depth}(C(\pi))$

$\mathrm{rt}(G) := \max_{\pi} \mathrm{rt}(G,\pi)$

So, the routing number of a graph is a functional measure of connectivity that tells us how slow permuting on it can be.



Quantum Routing

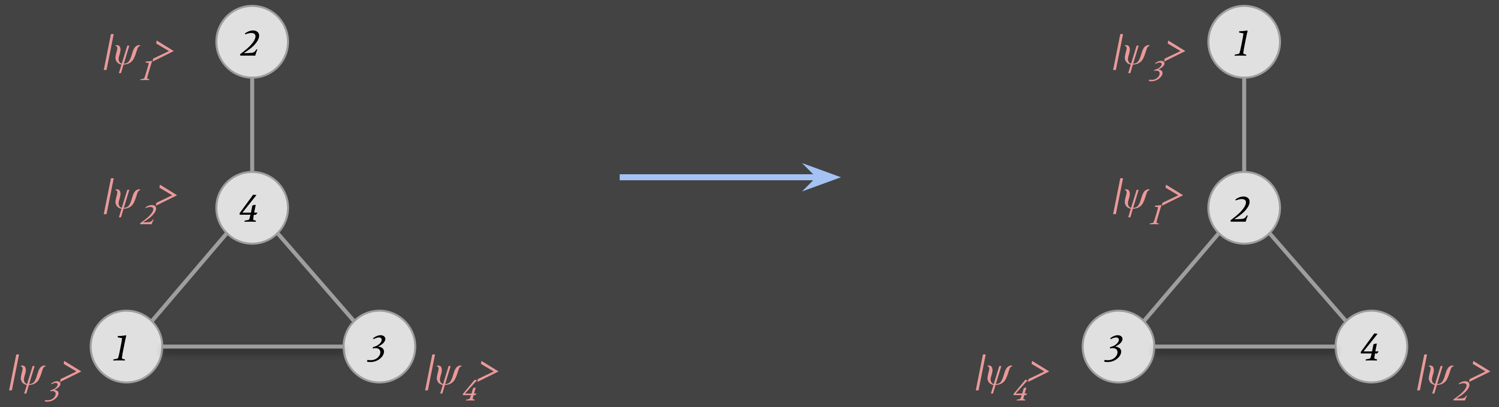
Now, suppose every node *i* contains a qubit in an unknown state $|\psi_i>$.

Item 1

Goal: Permute qubits on a graph with limited connectivity, i.e., implement

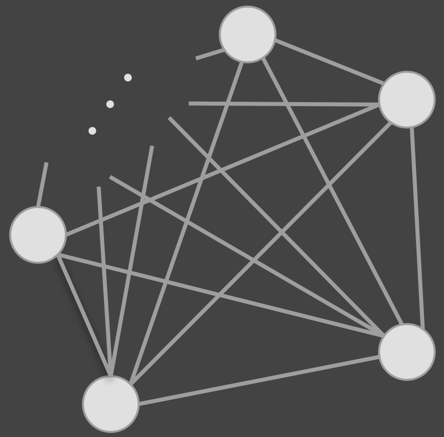
$$U_\pi: \; |\psi_1> \otimes |\psi_2> \otimes \ldots \otimes |\psi_N> \; \mapsto |\psi_{\pi(1)}> \otimes |\psi_{\pi(2)}> \otimes \ldots \otimes |\psi_{\pi(N)}>$$

via operations that act on single nodes or pairs of nodes connected by an edge.

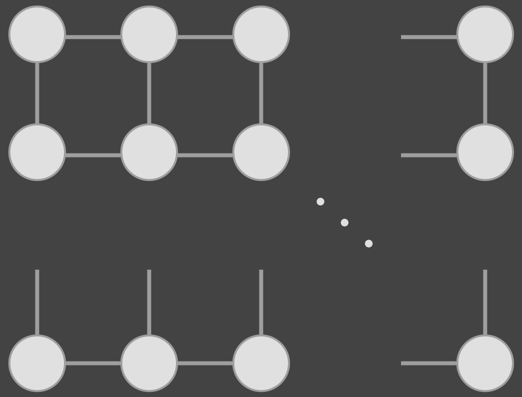


Why study routing?

Because most algorithms assume this:



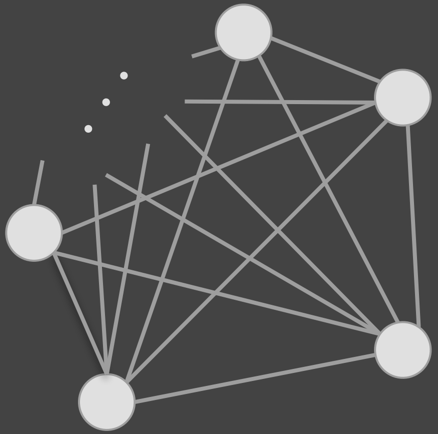
But most architectures look like this:



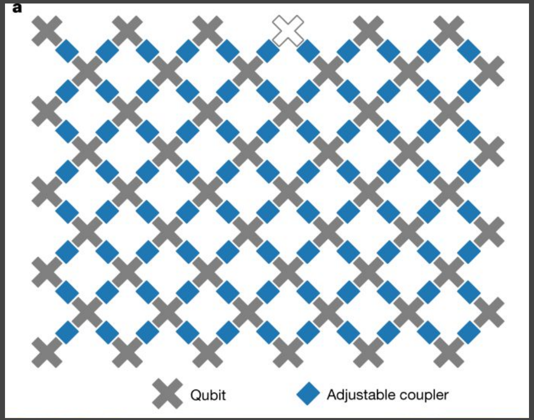
Therefore, it is usually necessary to permute distant qubits between steps of the algorithm.

Why study routing?

Because most algorithms assume this:



But most architectures look like this:

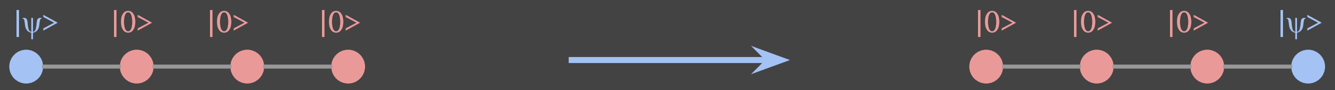


"Sycamore" (Google)

Therefore, it is usually necessary to permute distant qubits between steps of the algorithm.

Why study routing?

Most problems of state transfer in physical systems are restrictions of the routing problem.

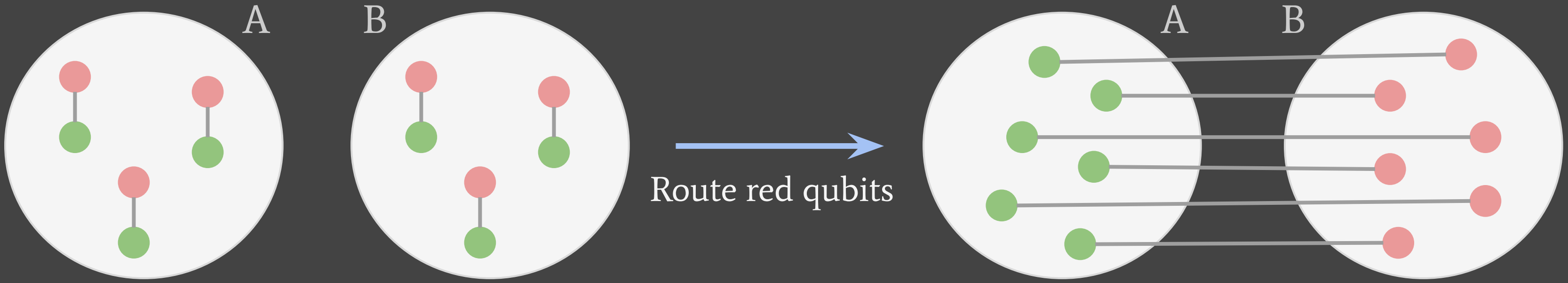


Extra assumptions (usually):

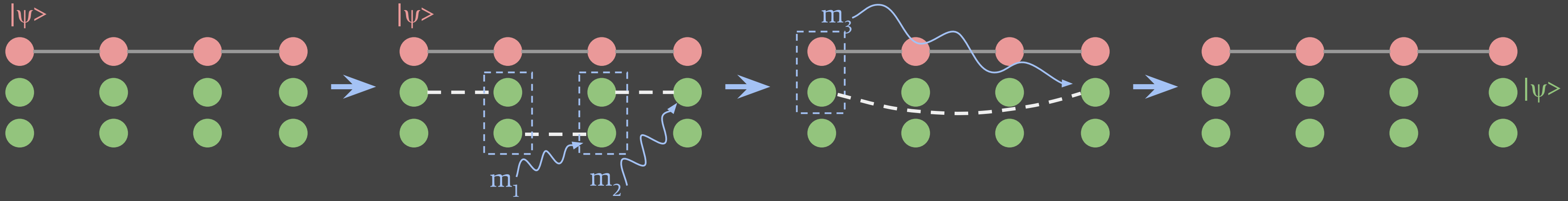
- Known initial state
- Fixed permutation

Why study routing?

Routing is one of the easiest ways to distribute a known amount of (bipartite) entanglement:



Conversely, routing can be achieved using entanglement via, e.g., the quantum repeater:



Is it possible to beat SWAP-based routing?

Routing of qubits is most commonly achieved using SWAP gates acting on neighboring nodes.

But in the quantum setting, a richer array of operations is available. It makes sense to study routing models with more power that might beat SWAP-based protocols.

Various models of quantum routing can be considered based on the allowed operations:

- SWAP gates (“classical”)
- 2-qubit gates (unitary)
- 2-local interactions (Hamiltonian)
- Measurement, classical communication (LOCC)

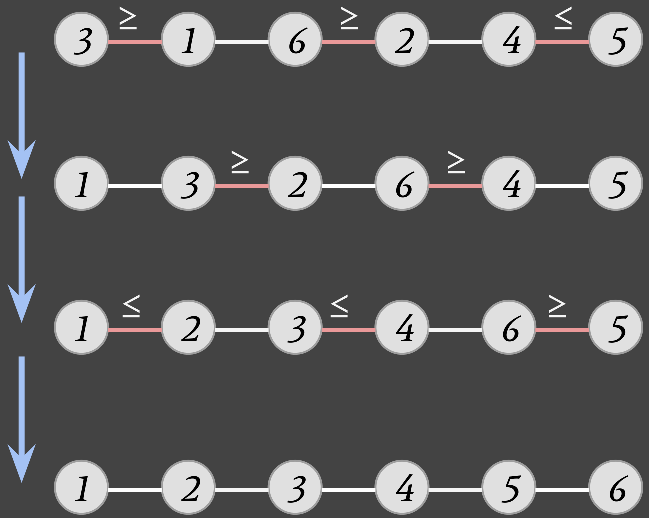
(+ ancilla)

SWAP-based routing on the path

The best known routing algorithm for the path is known as Odd-Even Sort (OES) and takes time N. This is classically optimal due to the diameter bound.

In fact quantum (non-LOCC) routing on the path must also scale as N.

However, we can hope for a constant factor speedup: T = cN in the worse case, where c < 1.



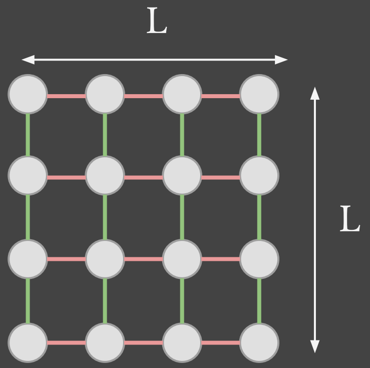
SWAP-based routing on the grid

Routing algorithms often use path routing as a subroutine (see, e.g. [1])

Example: 2D Grid

Naive algorithm: $O(L^2)$

OES-based routing: $O(L)$



Idea: Route rows in parallel, then columns, and then the rows again. (Why does this work?)

[1]: arXiv:1902.09102

Part II: Routing with fast reversal

Fast reversal on a path

In [2], we showed that the reversal permutation can be perfectly implemented in time N/3 using a time-independent Hamiltonian.

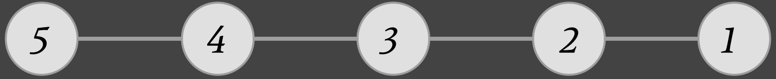
(The best SWAP-based protocol takes time N.)

Reversal = $\exp(iH^*(N+1)\pi/4) = \exp(iH^*(N+1)/3^*t_{SWAP})$

A time-dependent was protocol shown previously in [3].

Reversal = $\prod_{layer=1,\ldots,N+1} \exp(i\pi/4 \sum_i Z_i) \cdot \exp(i\pi/4 \sum_i X_i X_{i+1})$

P_5 :

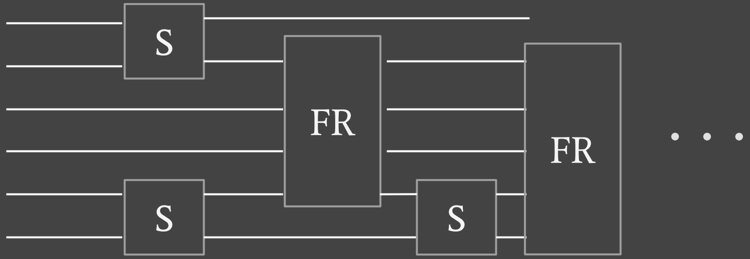


[2]: arxiv:2003.02843
[3]: arxiv:quant-ph/0505122

Routing via fast reversal (FR)

It is likely hard to design fast quantum protocols for every possible permutation. However, we already have a fast protocol for a *specific* permutation, namely, reversal.

Let's adopt a hybrid approach where we ask whether this fast primitive can be used by a classical algorithm to achieve fast routing. At the end we may have a circuit that looks like this:

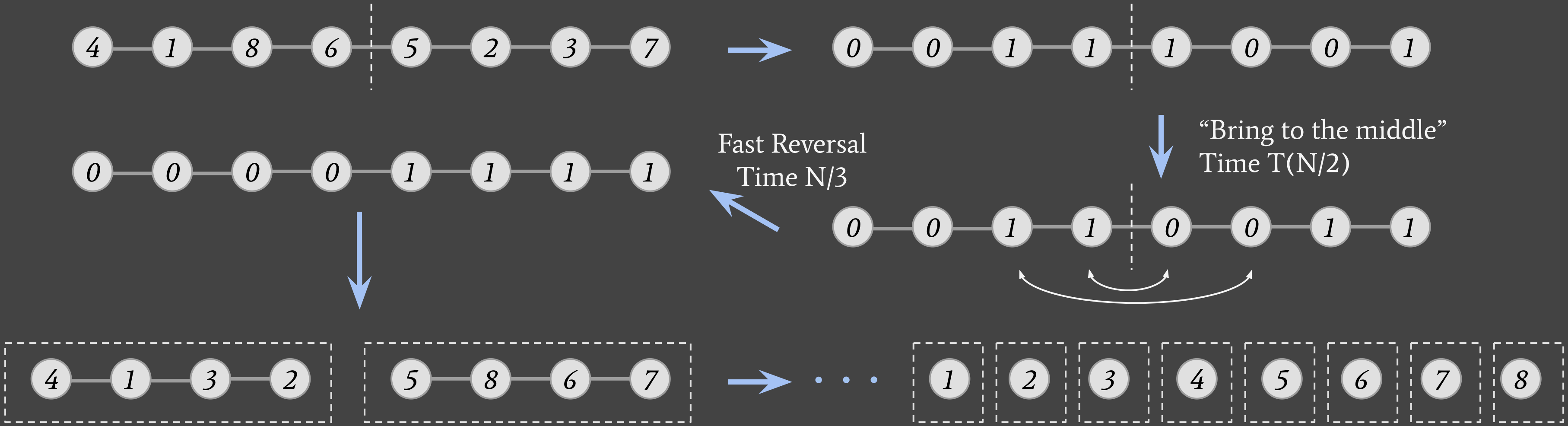


The goal then is, given any input permutation, to implement the permutation on an arbitrary state on the chain using FR. (Ideally, faster than OES.)

Divide-and-Conquer and Binary Sorting

Label nodes routing to the left half by ‘0’, and nodes routing to the right half by ‘1’. Sorting the string divides the problem divides into two sub-problems!

Apply binary sort on every sub-chain recursively until you have chains of length 1. This achieves routing on the original chain.

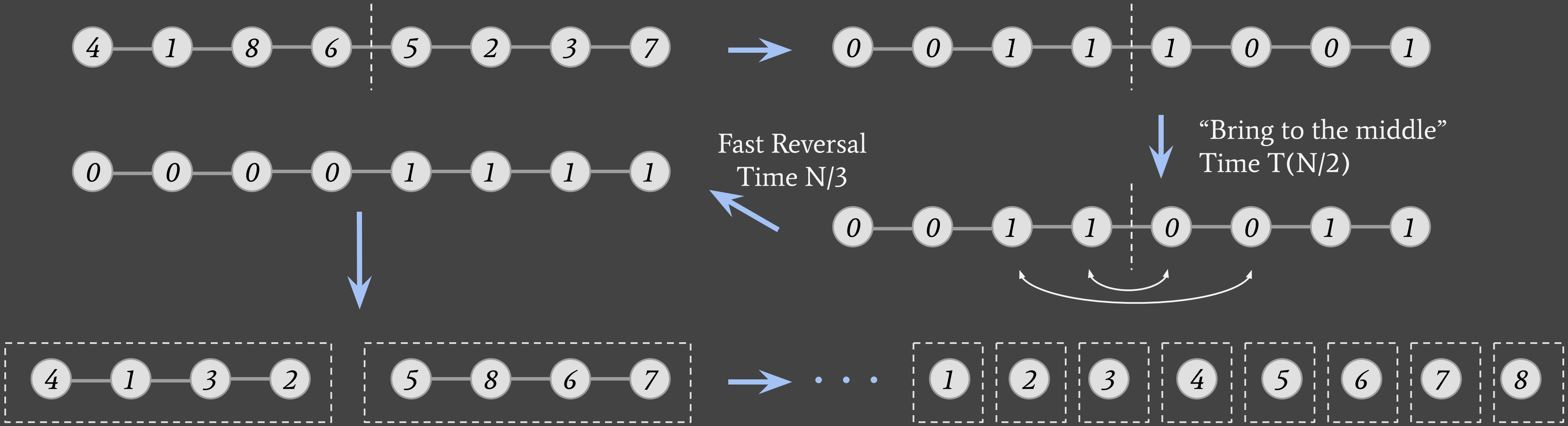


Binary Sort

So, binary sort takes time $T_{BS}(N) \leq T_{BS}(N/2) + N/3 = N/3 + N/6 + N/12 + \ldots = 2N/3$

The full algorithm uses binary sort recursively. Therefore, the total routing time is

$T(N) = T_{BS}(N)+T_{BS}(N/2)+ \ldots \leq 2N/3*(1+½+¼+ \ldots) = 4N/3$ .... Too slow :(



“Tripartite” Binary Sort (TBS)

Can we fix this? Well, yes! The middle reversal is the problem - it’s not “doing enough” for the time it takes. So, let’s make it more useful by dividing the chain into three parts instead of two.

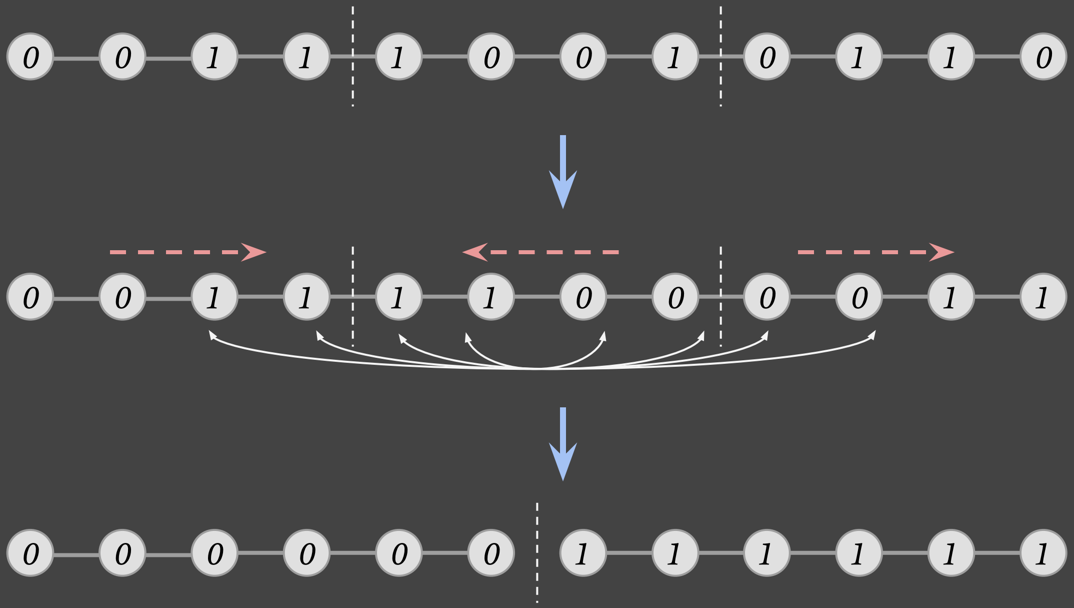
$T_{TBS}(N) \leq T_{TBS}(N/3) + N/3$

$= N/3(1+⅓+\ldots) = N/2$

So,

$T(N) = T_{TBS}(N)+T_{TBS}(N/2)+\ldots.$

$\leq\ N/2^*(1+½+¼+\ldots) = N\ \ (!)$



Worst-case performance

An upper bound of N is still not sufficient, since we know that the SWAP-based routing number for the path is N. But in fact, we can show the following:

Theorem: For all bit strings b of length n, $T_{TBS}(b) \leq (½-\varepsilon)N + O(\log(N))$ where ε is a small, positive constant.

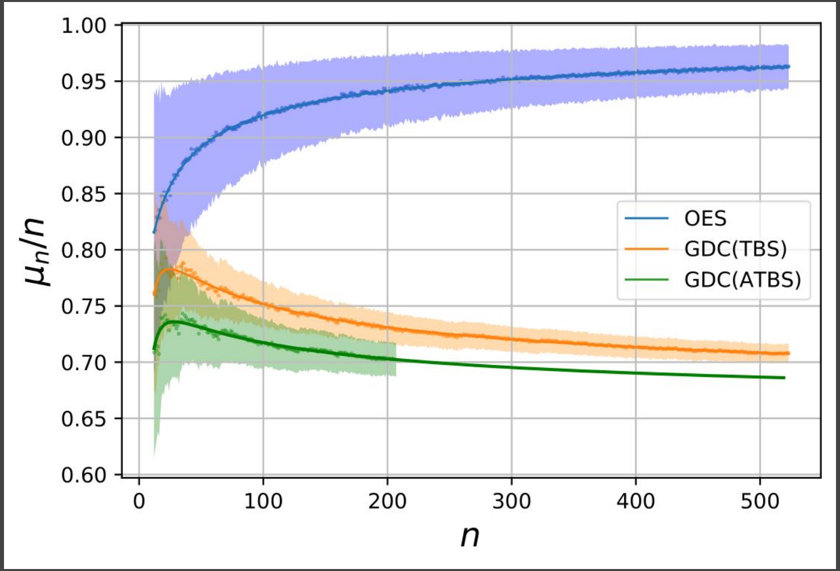
$\Rightarrow$ Routing using TBS beats any SWAP-based protocol! (As far as we know, this is the first known quantum speedup for the routing problem.)

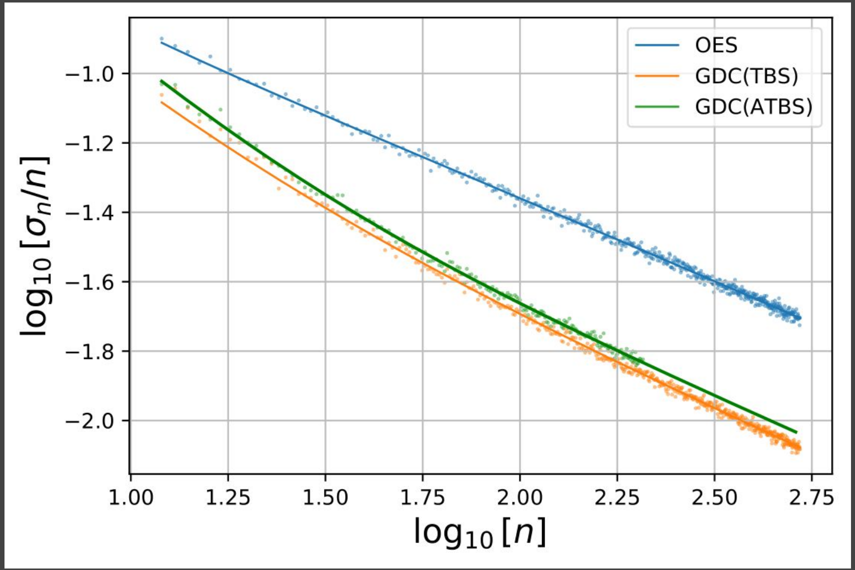
But does TBS offer any practical advantage?

- Is it faster on average? By how much?
- Is it as implementable as SWAP?

Average-case performance

We compare OES vs TBS vs Adaptive TBS for random permutations as a function of length.



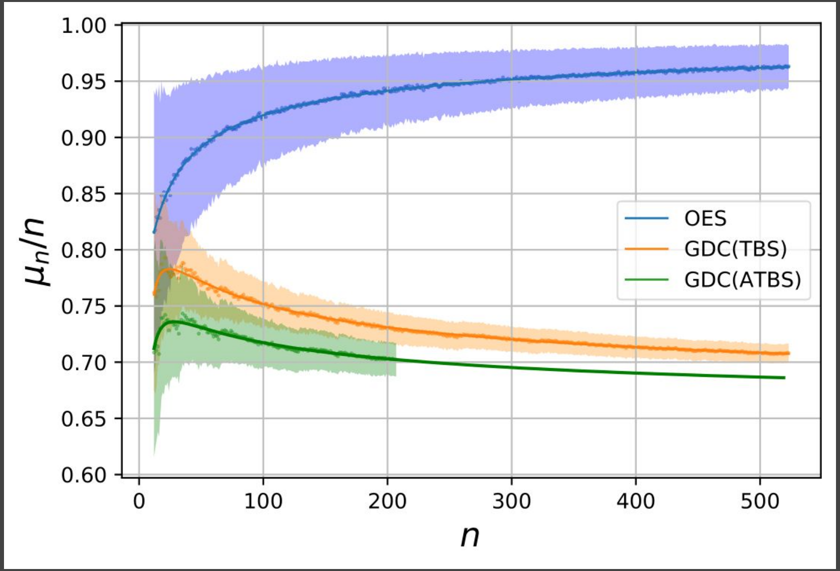


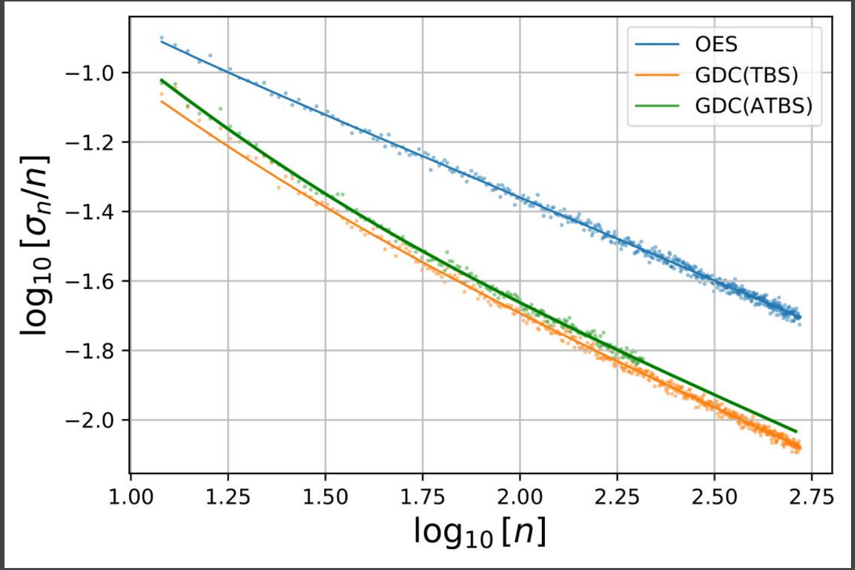
Numerical fits suggest that TBS-based routing takes time 2N/3 + O(√N) on average….

Average-case performance

Theorem: The average-case routing time of GDC(TBS) is $2N/3 + O(n^\alpha)$, where $\alpha \in (½ , 1)$.

$\Rightarrow$ On average, TBS-based (quantum) routing is 66% faster than SWAP-based routing :)





Implementability

SWAP(a,b) = CNOT(a,b) · CNOT(b,a) · CNOT(a,b). That's three entangling operations.

Therefore, a reversal on N qubits using OES would use ~$3N^2/2$ entangling gates classically.

On the other hand, a fast reversal can be implemented via Mølmer-Sørensen gates:

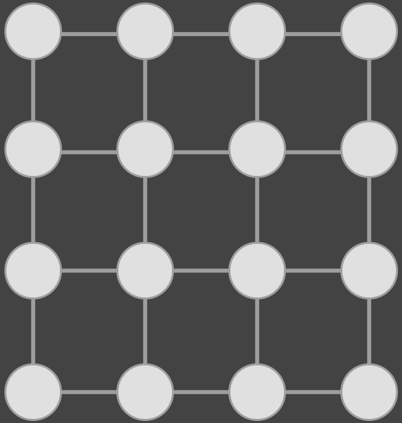
$$\text{Reversal} = \prod_{\text{layer}=1,\dots,N+1} \exp(i\pi/4 \textstyle\sum_i Z_i) \cdot \exp(i\pi/4 \textstyle\sum_i X_i X_{i+1})$$

This is only N^2 entangling gates. Depending on the native gates in the hardware, fast reversal could be no harder to implement than SWAP-based routing (and likely faster).

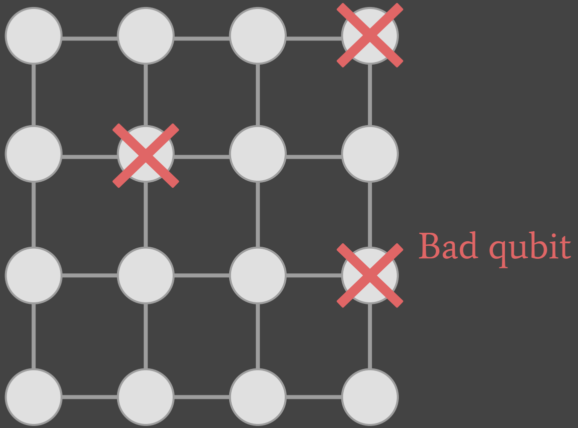
Part III: Routing with Defects

Defects

Earlier, I said architectures look like this:



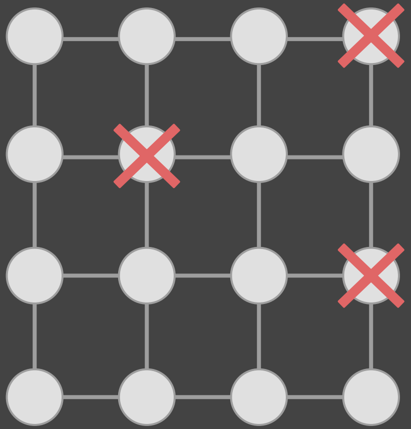
But *actually*, they're more like this:



How do you route on a defective grid?

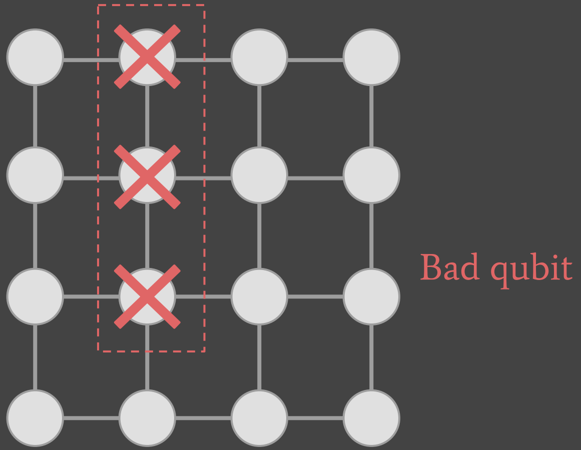
Defect models

Random defects (each qubit fails with probability p)



- Defective edges, etc.

Correlated defects, or defective regions



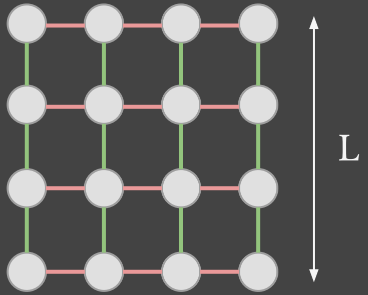
Routing on the grid

Routing algorithms often use path routing as a subroutine (see, e.g. [1])

Example: 2D Grid

Naive algorithm: $O(L^2)$

OES-based routing: $O(L)$



Idea: Route rows in parallel, then columns, and then the rows again. (Why does this work?)

[1]: arXiv:1902.09102

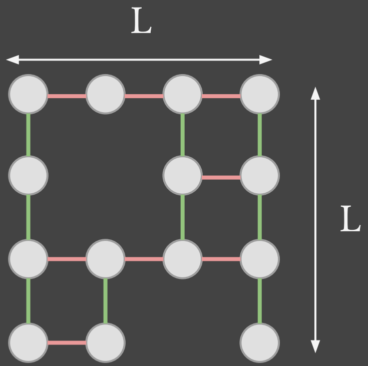
Routing on the defective grid?

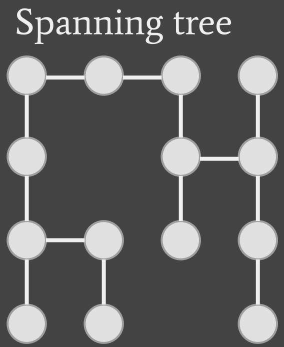
Routing algorithms often use path routing as a subroutine (see, e.g. [1])

Example: Defective Grid

Naive algorithm: $O(L^2)$

OES-based routing: $\Omega(L)$, O(??)





Possible questions:

- On a scale of $O(L)$ to $O(L^2)$, what is the routing time of a defective grid?
- Is routing on (random) defective grids robust in the failure probability p?
- What happens when you allow quantum primitives such as fast reversal?

Thanks!