Quantum Distributed Algorithms for Approximate Steiner Trees and Directed Minimum Spanning Trees

Phillip Kerger[1,2,3], David Bernal Neira[1,2,4], Zoe Gonzalez Izquierdo[1,2,] Eleanor Rieffel[1]

1: NASA Quantum Artificial Intelligence Laboratory
2: USRA Research Institute for Advanced Computer Science
3: Department of Applied Mathematics, Johns Hopkins University
4: Department of Chemical Engineering, Purdue University

Based on work with…

Phillip Kerger[1,2,3]

David Bernal Neira[1,2]

Zoe Gonzalez Izquierdo[1,2]

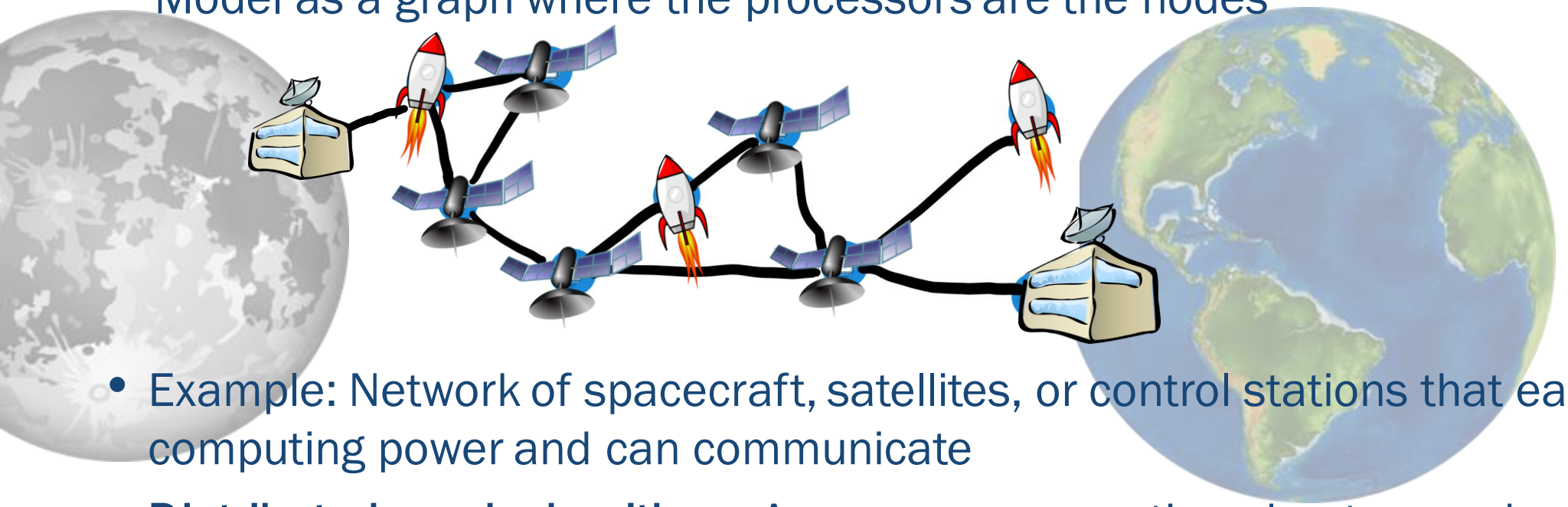
Eleanor Rieffel[1]

"Mind the $\tilde{O}$: asymptotically better, but still impractical, quantum distributed algorithms", https://arxiv.org/abs/2304.02825

1: NASA Quantum Artificial Intelligence Laboratory
2: USRA Research Institute for Advanced Computer Science
3: Department of Applied Mathematics, Johns Hopkins University

Introduction: Setting

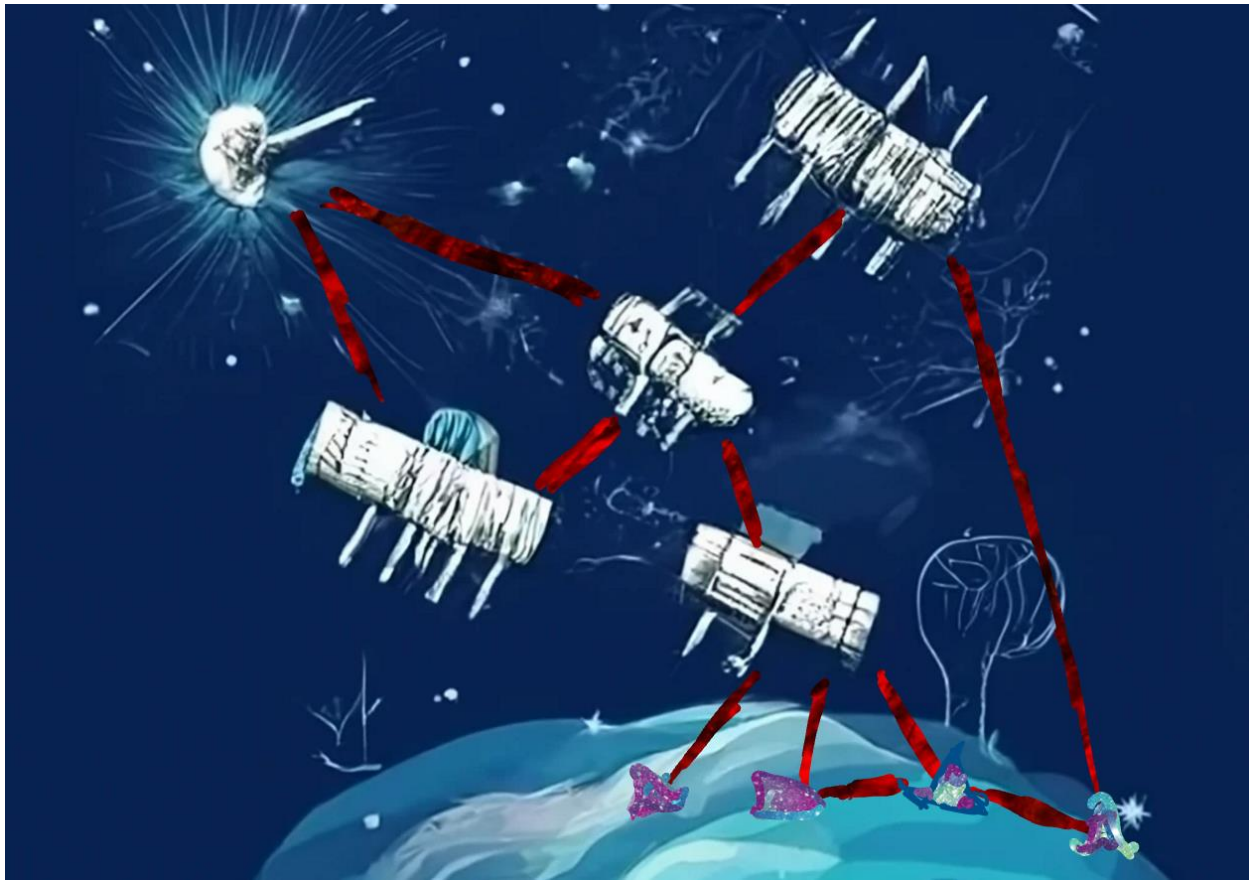
- **Distributed Computation:** Network of Multiple Processors that communicate
- Model as a graph where the processors are the nodes



- Example: Network of spacecraft, satellites, or control stations that each have computing power and can communicate
- **Distributed graph algorithms:** Answer some question about a graph whose nodes are the processors, where each node has local information

Introduction: Setting

- **CONGEST-CLIQUE Model:**
  1. Computation happens in *rounds*
     -(compute, communicate, compute, …)
  2. **Congested:** Communication limited by message size O(log(n)) **bits,**
     n being the number of nodes in the graph
     →one message = info of one edge of graph
  3. Unlimited local computation at each node
  4. Nodes have local information about graph
- **Goal:** Answer a question about the graph
- **Quantum Model:** Qubit Communication

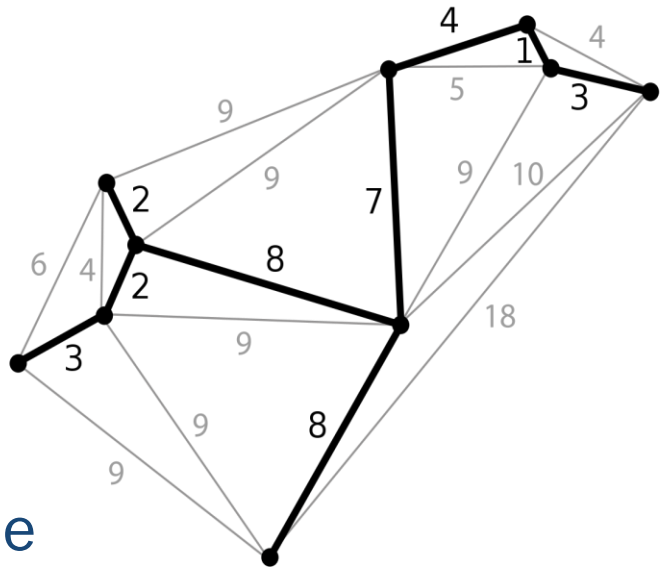


Thanks to craiyon.com for generating the image

Introduction: Models

- Initially: Given an input graph, each node knows its own ID and the (weights of) edges it is incident upon
- Assume ID's are 1 to n → log(n) bits to encode a node ID

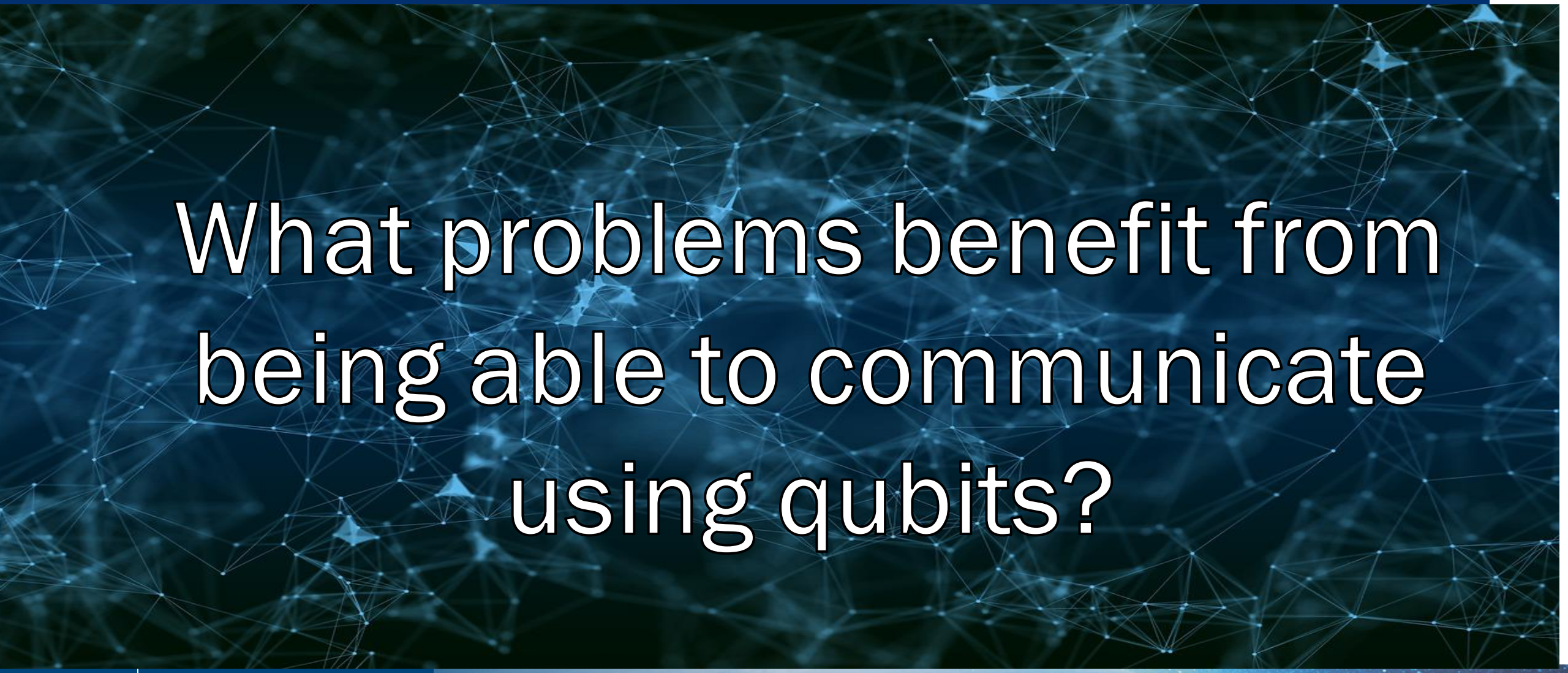
- Answer a question about that graph in as few rounds as possible
  - Ex: Spanning trees, subgraph detection, shortest paths…



A minimum spanning tree (black) for the given graph (grey)
source: https://en.wikipedia.org/wiki/Minimum_spanning_tree

- Quantum versions: Take CONGEST or CONGEST-CLIQUE, but allow messages to consist of O(log(n)) **qubits**

Core Research Question:



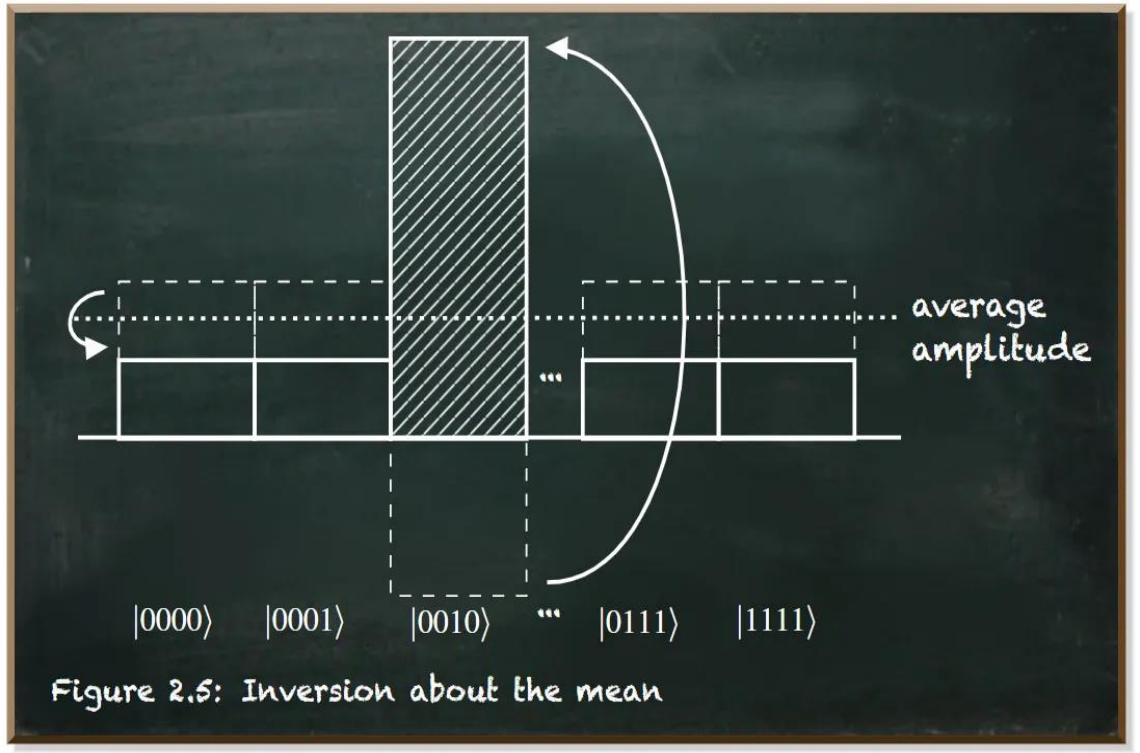
Distributed Grover Search

First, a useful tool...

Grover Search

- Given an unstructured search over search space X, i.e. a black-box $f(y) \to \{0,1\}$ and we want y in X: $f(y) = 1$, Grover search does this in roughly $\sqrt{|X|}$ evaluations of f.
- A classical algorithm needs $|X|$ evaluations

→ quadratic speedup



Grover Search: Distributed

Distributed Implementation of Grover's Algorithm:

- Suppose: a set X of inputs, and
  a node u can evaluate a function $g: X \to \{0,1\}$ in r rounds.
  There exists a quantum distributed algorithm such that u finds an element y giving $g(y) = 1$ using $O\left(r\sqrt{|X|}\right)$ rounds.

- Intuition: Evaluate g distributedly, with inputs in superposition, apply the same ideas as Grover search

- What can this help with?

Triangle Finding

Make use of Distributed Grover Search for finding Triangles!

Idea:

- Triangles live in $V \times V \times V$, so partition that space
- Have each node search an assigned partition for triangles
- Do this via sets $\boldsymbol{V_1} \times \boldsymbol{V_2} \times \boldsymbol{V_3}$, where $|\boldsymbol{V_1}| = |\boldsymbol{V_2}| = n^{1/4}, |\boldsymbol{V_3}| = \mathrm{n}^{1/2}$, and they each $\boldsymbol{w} \in V_1$ or V_2 contains $n^{3/4}$ nodes, and each $\boldsymbol{w} \in \boldsymbol{V_3}$ contains $n^{1/2}$ nodes.

- So $\boldsymbol{V_1} \times \boldsymbol{V_2} \times \boldsymbol{V_3}$ has n elements that are each sets of nodes → each node gets one
- Making use of fast routing, the bottleneck becomes searching through the elements of $\boldsymbol{V_3}$ → Grover can do it in $n^{1/4}$ instead of $n^{1/2}$

Distance Products

Definition 3.2. The *distance product* between two $n \times n$ matrices A and B, also sometimes referred to as the min-plus or tropical product, is defined as:

$$(A \star B)_{ij} = \min_k \{A_{ik} + B_{kj}\}. \tag{3.1}$$

- If W is the adjacency matrix of a graph, then the ij entry of $W \star W$ gives the shortest 2-hop path from node i to node j
- W^n contains shortest n-hop distances , so contains all shortest path distances (exponent w.r.t. $\star$) → take $\log(n)$ squares:

$$W^{2,\star} = W \star W, \quad W^{4,\star} = \left(W^{2,\star}\right)^{2,\star}, \ldots, \quad W^{2^{\lceil \log n \rceil},\star} = \left(W^{2^{\lceil \log n \rceil - 1},\star}\right)^{2,\star}$$

Distance Products via Triangles

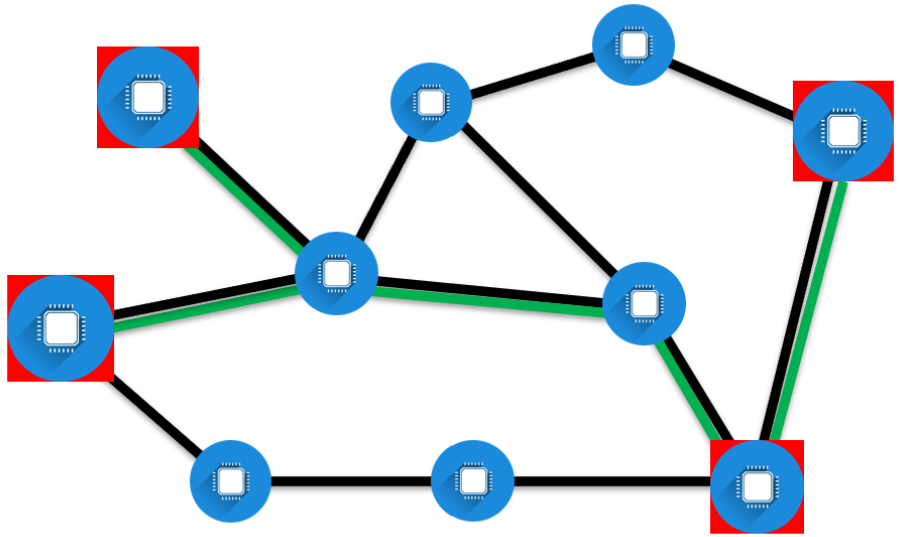
Proposition 3.3. If $FindTriangleEdges^{-}$ on an n-node integer-weighted graph $G = (V, E, W)$ can be solved in $T(n)$ rounds, then the distance product $A \star B$ of two $n \times n$ matrices A and B can be computed in $T(3n) \cdot \lceil \log_2(\max_{v,z \in G}\{\min_{u \in V}\{A_{vu} + B_{uv})\}\}\rceil$ rounds.

Proof idea:

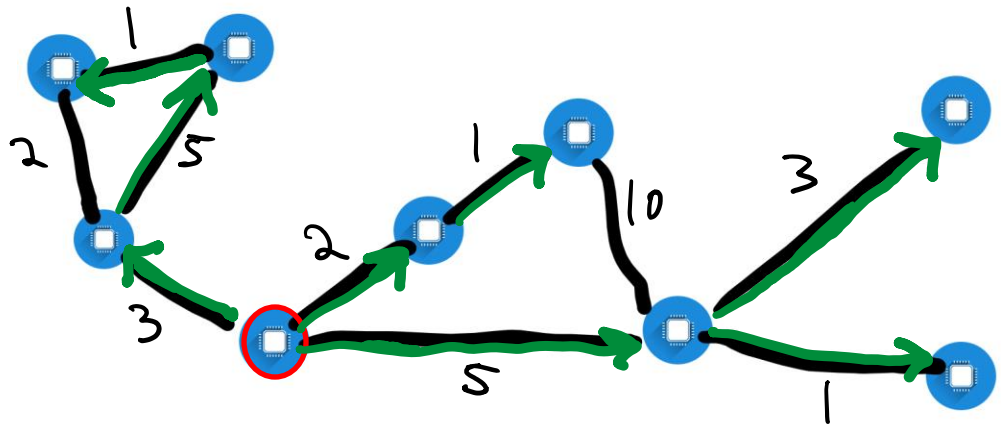
- Use negative triangle detection to do a binary search for distances
- Observation: $A_{vu} + B_{uz} < d$ exactly whenever adding an edge of weight $-d$ from z to v gives a negative triangle → do a binary search like this!

Problems

- Answer a question about the graph in as few rounds as possible
  - **Steiner Tree, (Directed) Minimum Spanning Tree,** shortest paths, max clique, Hamiltonian cycle…



A **Steiner Tree** (green) for red terminals



A **Directed Minimum Spanning Tree** (green) rooted at the red node

Contributions:

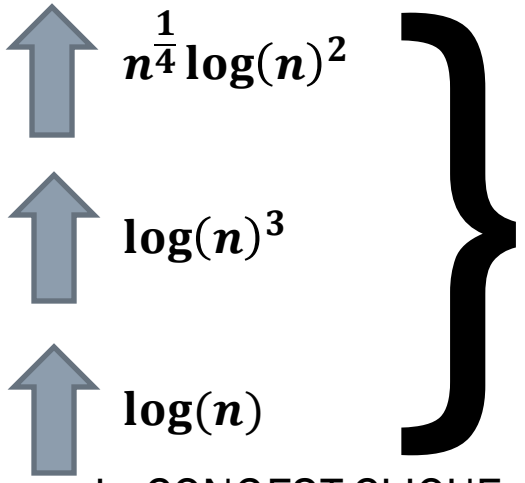
- Algorithms in Quantum CONGEST-CLIQUE for
  - **Approximate Steiner Trees**
  - **Directed Minimum Spanning Trees**
- Asymptotically fewer rounds than any known classical algorithm

  $\rightarrow \tilde{O}(n^{1/4})$ versus $\tilde{O}(n^{1/3})$
- $\tilde{O}$-notation hides constants and logarithms, commonly used for complexities

- Exact complexity analysis of quantum *and* classical algorithms reveals: improvements needed for ***both***

Algorithmic Recipe & Complexity

1. Distributed Grover Search
   helps with…
2. Triangle Finding
   helps with…
3. Distance Products
   helps with…
4. Shortest Paths and Routing Tables
   helps with…
5. Steiner and Directed Minimum Spanning Trees!



$c \cdot n^{\frac{1}{4}}\log(n)^6$ rounds

In CONGEST CLIQUE, can solve anything in n rounds:
To be practical, need

$$800 \cdot n^{\frac{1}{4}}\log(n)^6 < n$$

for which

$$n > 10^{18}$$

is required!! 10^{11} for classical $\tilde{O}\left(n^{\frac{1}{3}}\right)$ counterpart.

Takeaways:

- Practicality of algorithms can be massively impacted by $\tilde{O}$-obscured terms, much more so than by regular O-notation
- As we draw closer to implementations being possible, algorithmists should be encouraged to provide *exact* complexities
  - Larger burden for quantum due to hardware?
- Practitioners should be wary of $\tilde{O}$ to decide what algorithms to use, in classical too
- Bright side: Asymptotic speedups may bring ideas needed for practical faster algorithms

Main References

Elkin, M., Klauck, H., Nanongkai, D., & Pandurangan, G. (2012). *Can quantum communication speed up distributed computation?* arXiv. Retrieved from https://arxiv.org/abs/1207.5211 doi doi:10.48550/ARXIV.1207.5211

Izumi, T., & Gall, F. L. (2019, jul). Quantum distributed algorithm for the all-pairs shortest path problem in the CONGEST-CLIQUE model. In *Proceedings of the 2019 ACM symposium on principles of distributed computing*. ACM. Retrieved from https://arxiv.org/abs/1906.02456 doi: doi:10.1145/3293611.3331628

Izumi, T., Le Gall, F., & Magniez, F. (2020). *Quantum distributed algorithm for triangle finding in the congest model.* Schloss Dagstuhl - Leibniz-Zentrum für Informatik. Retrieved from https://drops.dagstuhl.de/opus/volltexte/2020/11884/ doi: doi:10.4230/LIPICS.STACS.2020.23

Saikia, P., & Karmakar, S. (2019). *Distributed approximation algorithms for steiner tree in the congested clique.* arXiv. Retrieved from https://arxiv.org/abs/1907.12011 doi: doi:10.48550/ARXIV.1907.12011

Paper link:

Images: craiyon.com (AI image generator), credit to pixabay.com (copyright free images)

Full Steiner Tree Approximation Algorithm

Step 1 - APSP and Routing Tables: Solve the APSP problem and add an efficient routing table scheme via triangle finding in $\tilde{O}(n^{1/4})$ rounds

Step 2 - Shortest-path Forest: Construct a shortest-path forest (SPF), where each tree consists of exactly one source terminal and the shortest paths to the vertices whose closest terminal is that source terminal. This step can be completed in one round and n messages

Step 3 - Weight Modifications: Modify the edge weights depending on whether they belong to a tree (set to 0), connect nodes in the same tree (set to ∞), or connect nodes from different trees (set to distance of shortest path between root terminals of the trees that use the edge).

Step 4 - Minimum Spanning Tree: Construct a minimum spanning tree (MST) on the modified graph in $O(1)$ rounds

Step 5 - Pruning: Prune leaves of the MST that are non-terminal nodes, since these are not needed for the Steiner Tree

Visualization