Two Graph Problems Related to Quantum Compiling: Qubit Routing and Parallel Token Swapping

Oktay Günlük

H. Milton Stewart School of Industrial and Systems Engineering

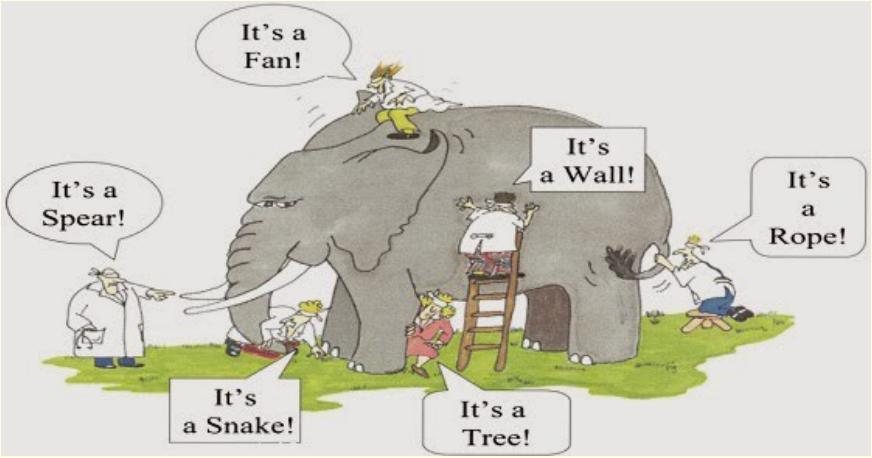
Georgia Tech

Rice University, Houston

November 2025

What is Quantum Computing?

What is Quantum Computing?



Answer: Depends on who you talk to...

Quantum Computing Pipeline:

Algorithm (high-level)

↓

Gate Decomposition

↓

Qubit Mapping & Routing

↓

Optimization (peephole)

↓

Error-Aware Compilation (hardware)

↓

Hardware Instructions (native gates)

↓

Execution (noisy but deterministic)

↓

Measuring Output (not deterministic)

Quantum Computing Pipeline:

Algorithm (high-level)

↓

Gate Decomposition

↓

Qubit Mapping & Routing

↓

Optimization (peephole)

↓

Error-Aware Compilation (hardware)

↓

Hardware Instructions (native gates)

↓

Execution (noisy but deterministic)

↓

Measuring Output (not deterministic)

Disclaimer: I know very little about most of the above

Quantum Algorithms as Unitary Operators

- Any quantum algorithm on n qubits is a unitary transformation U that acts on a 2^n-dimensional Hilbert space $\mathcal{H}_n = \mathbb{C}^{2n}$.
- A **universal gate set** can approximate any unitary to arbitrary accuracy.

  Examples:

  $$\{H, T, \mathrm{CNOT}\}, \quad \{H, S, T, \mathrm{CNOT}\}.$$

  - Some of these gates operate on a single qubit (H, S, T) (rotation)
  - Others involve pairs of qubits (CNOT)
- Hence, 1- and 2-qubit gates suffice for universality. Formally:

  $$U = \prod_{i \in I} U_i, \qquad U_i \text{ acts on at most two qubits.}$$

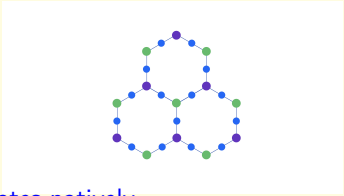
- 1-qubit gates can perform any local rotation, but they cannot create entanglement.
- Entanglement is necessary for quantum advantage.

Gate Decomposition and Practical Implications

Given a unitary transformation U, decompose it to simple gates:

$$U = \prod_{i \in I} U_i,$$

where each gate U_i involves at most 2 qubits.



Why is this needed in practice and implications:

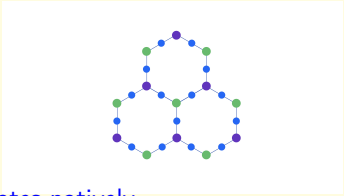
- Most physical devices can only execute 1- and 2-qubit gates natively.
- And these 2-qubit gates can be executed only if the associated qubits are physically adjacent on the (sparse) hardware graph.
- But these 2-qubit gates can involve arbitrary pairs of qubits.
- To execute the quantum algorithm, qubits need to be rearranged throughout the process

Gate Decomposition and Practical Implications

Given a unitary transformation U, decompose it to simple gates:

$$U = \prod_{i \in I} U_i,$$

where each gate U_i involves at most 2 qubits.



Why is this needed in practice and implications:

- Most physical devices can only execute 1- and 2-qubit gates natively.
- And these 2-qubit gates can be executed only if the associated qubits are physically adjacent on the (sparse) hardware graph.
- But these 2-qubit gates can involve arbitrary pairs of qubits.
- To execute the quantum algorithm, qubits need to be rearranged throughout the process

Sparse hardware graphs and SWAP operation

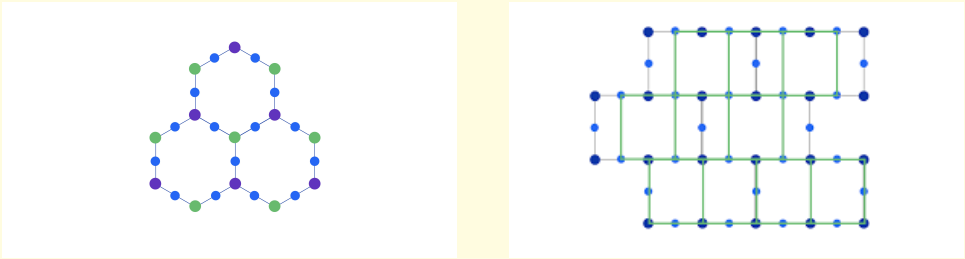


Figure: Heavy-hex lattice graph

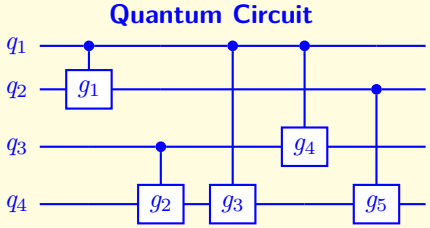
How to move qubits around:

- Adjacent qubits can be SWAPped (not the physical qubits but the information they have)
- This is implemented by using 3 CNOT gates
- SWAP operations are time consuming and qubit quality degrades over time,
- This in turn degrades the quality (error) of the output

Example

- Consider a quantum circuit that contains 4 qubits $Q = \{q_1, q_2, q_3, q_4\}$
- Consider a quantum circuit with the following five two-qubit gates:

  $g_1 = \{q_1, q_2\},\ g_2 = \{q_3, q_4\},\ g_3 = \{q_1, q_4\},\ g_4 = \{q_1, q_3\}$, and $g_5 = \{q_2, q_4\}$.
- If the gates are executed in this order, the circuit is executed correctly
- A gate can be executed after all dependencies are resolved

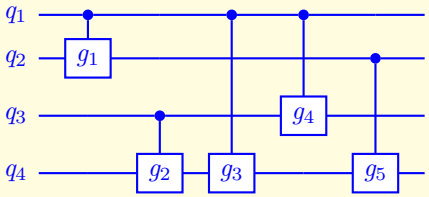


Example

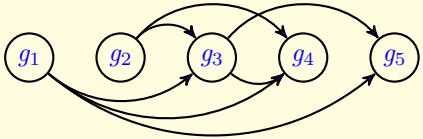
- Consider a quantum circuit that contains 4 qubits $Q = \{q_1, q_2, q_3, q_4\}$
- Consider a quantum circuit with the following five two-qubit gates:

  $g_1 = \{q_1, q_2\},\ g_2 = \{q_3, q_4\},\ g_3 = \{q_1, q_4\},\ g_4 = \{q_1, q_3\}$, and $g_5 = \{q_2, q_4\}$.
- If the gates are executed in this order, the circuit is executed correctly
- A gate can be executed after all dependencies are resolved

Quantum Circuit



Gate Dependency Graph (DAG)



- Some gates can be grouped and executed simultaneously:

$$G_1 = \{g_1, g_2\}, \quad G_2 = \{g_3\}, \quad G_3 = \{g_4, g_5\}$$

A building block for quantum algorithms

Definition

The **Quantum Fourier Transform** is the linear transformation

$$|x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i x y/N} |y\rangle, \qquad N = 2^n.$$

It is the quantum analogue of the discrete Fourier transform.

- Key ingredient in several quantum algorithms, including Shor's factoring algorithm.
- Can be implemented using 1- and 2-qubit gates
- An n-qubit QFT contains $\Theta(n^2)$ two-qubit gates.
- Many of these interactions occur between qubits that are far apart, making QFT a natural benchmark for qubit routing algorithms.

QFT is often used as a benchmark for qubit routing because routing overhead can dominate its execution cost on sparse hardware.

Qubit assignment and routing

(Joint work with Giacomo Nannicini, Lev S. Bishop, and Petar Jurcevic)

Problem Definition

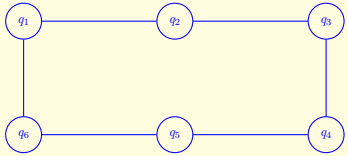
Given:

1. A connected hardware graph $H = (V, E)$, where
   - Nodes correspond to physical qubits
   - Edges correspond to pairs of qubits that can implement two-qubit gates.
2. A set of logical qubits Q and sequence of gate groups $G = \{G^1, G^2, \ldots, G^m\}$, where
   - $G^t = \{g_1^t, g_2^t, \ldots\}$ is a collection of 2-qubit gates that needs to be implemented at time step (or, depth) $t \in T = \{1, \ldots, m\}$, and
   - Each gate $g_l^t = \{p, q\}$ is a pair of qubits $p, q \in Q$ (that need to interact at time t)
   - If $G^t \neq \emptyset$ then t is an active gate layer.
   - If $G^t = \emptyset$ then t is a swap gate layer (inserted to allow rearranging qubits).

Objective: Use as few swap gate layers as possible while making sure that logical qubits $g_l^t = \{p, q\}$ are located in adjacent phsical qubits in G at time t for all $g_l^t \in G^t$.

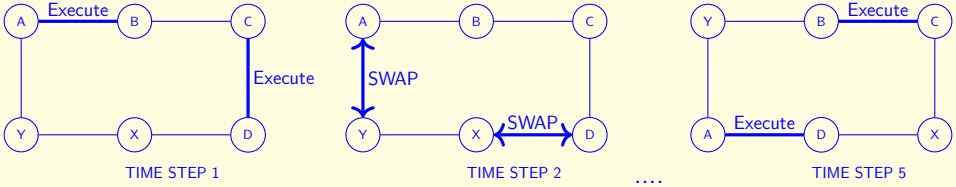
Example:

Hardware graph:



Gate groups: $G_1 = \Big\{ \underbrace{\{A, B\}}_{g_1}, \underbrace{\{C, D\}}_{g_2} \Big\}$ $\underbrace{G_2 = G_3 = G_4 = \emptyset}_{\text{SWAP layers}}$ $G_5 = \Big\{ \underbrace{\{A, D\}}_{g_3}, \underbrace{\{B, C\}}_{g_4} \Big\}$

Qubit Assignment and Routing:



- Time steps 3 and 4 are not used and circuit is executed in 3 time steps.

Optimization Problem:

- Objective: Minimize the number of SWAP layers used
  - These layers are used if there is any SWAP activity happens in that time step.
- Keep track of physical location of each logical qubit at all time steps
- Make sure that qubit pairs that are involved in gate group G_t are adjacent at time t
- Move qubits around to rearrange them
  - Two qubits can SWAP their locations at time t if they are next to each other, and,
  - (1) They are involved in a gate at time t, or, (2) Neither is involved in any gate at time t

Model: Turn directed hardware graph to an undirected one: $H = (V, E) \rightarrow D = (V, A)$

- Turn each edge $\{i, j\}$ of the hardware graph into 2 directed edges: (i, j) and (j, i).
- A qubit q located at node i can move along (i, j) if the qubit p located at j moves along (j, i).
- Each (undirected) gate $\{p, q\}$ is turned into a directed gate (p, q) which can be executed if
  - For some directed edge (i, j) we have p at node i and q at node j

Optimization Problem:

- Objective: Minimize the number of SWAP layers used
  - These layers are used if there is any SWAP activity happens in that time step.
- Keep track of physical location of each logical qubit at all time steps
- Make sure that qubit pairs that are involved in gate group G_t are adjacent at time t
- Move qubits around to rearrange them
  - Two qubits can SWAP their locations at time t if they are next to each other, and,
  - (1) They are involved in a gate at time t, or, (2) Neither is involved in any gate at time t

Model: Turn directed hardware graph to an undirected one: $H = (V, E) \rightarrow D = (V, A)$

- Turn each edge $\{i, j\}$ of the hardware graph into 2 directed edges: (i, j) and (j, i).
- A qubit q located at node i can move along (i, j) if the qubit p located at j moves along (j, i).
- Each (undirected) gate $\{p, q\}$ is turned into a directed gate (p, q) which can be executed if
  - For some directed edge (i, j) we have p at node i and q at node j

Binary IP Formulation:

$$\min \sum_{t \in T^{\text{SWAP}}} z^t \qquad \leftarrow \; [z^t = 1 \quad \text{if swap layer } t \text{ used}]$$

$$\text{s.t.} \sum_{j \in V} w^t_{q,j} = 1 \qquad \forall q \in Q,\ \forall t \in T \qquad \text{(Loc 1)}$$

$$\sum_{q \in Q} w^t_{q,j} = 1 \qquad \forall j \in V,\ \forall t \in T \qquad \text{(Loc 2)}$$

$$w^t_{q,i} = x^t_{q,i,i} + \sum_{k:(i,k) \in A} x^t_{q,i,k} \qquad \forall q \in Q,\ i \in V,\ \forall t \in T \setminus \{m\}, \qquad \text{(Route 1)}$$

$$w^t_{q,i} = x^{t-1}_{q,i,i} + \sum_{k:(k,i) \in A} x^{t-1}_{q,k,i} \qquad \forall q \in Q,\ \forall i \in V,\ \forall t \in T \setminus \{1\} \qquad \text{(Route 2)}$$

$$x^t_{p,i,j} = x^t_{q,j,i} \qquad \forall t \in T \setminus \{m\},\ (p,q) \in G^t,\ (i,j) \in A \qquad \text{(Swap 1)}$$

$$\sum_{q \notin Q^t} x^t_{q,i,j} = \sum_{p \notin Q^t} x^t_{p,j,i} \qquad \forall t \in T \setminus \{m\},\ \forall (i,j) \in A \qquad \text{(Swap 2)}$$

$$\sum_{(i,j) \in A} y^t_{(p,q),(i,j)} = 1 \qquad \forall t \in T,\ \forall (p,q) \in G^t \qquad \text{(Gate 1)}$$

$$\{w^t_{p,i}, w^t_{q,j}\} \geq y^t_{(p,q),(i,j)} \geq \{0, w^t_{p,i} + w^t_{q,j} - 1\} \qquad \forall t \in T,\ \forall (p,q) \in G^t,\ \forall (i,j) \in A \qquad \text{(Gate 2)}$$

$$\sum_{(i,j) \in A} x^t_{q,i,j} \leq z^t \qquad \forall q \in Q,\ \forall t \in T^{\text{SWAP}} \qquad \text{(Count)}$$

Binary IP Formulation:

$$\min \sum_{t \in T^{\text{SWAP}}} z^t \qquad \leftarrow \; [z^t = 1 \quad \text{if swap layer } t \text{ used}]$$

$$\text{s.t.} \sum_{j \in V} w^t_{q,j} = 1 \qquad \forall q \in Q,\ \forall t \in T \qquad \text{(Loc 1)}$$

$$\sum_{q \in Q} w^t_{q,j} = 1 \qquad \forall j \in V,\ \forall t \in T \qquad \text{(Loc 2)}$$

$$w^t_{q,i} = x^t_{q,i,i} + \sum_{k:(i,k) \in A} x^t_{q,i,k} \qquad \forall q \in Q,\ i \in V,\ \forall t \in T \setminus \{m\}, \qquad \text{(Route 1)}$$

$$w^t_{q,i} = x^{t-1}_{q,i,i} + \sum_{k:(k,i) \in A} x^{t-1}_{q,k,i} \qquad \forall q \in Q,\ \forall i \in V,\ \forall t \in T \setminus \{1\} \qquad \text{(Route 2)}$$

$$x^t_{p,i,j} = x^t_{q,j,i} \qquad \forall t \in T \setminus \{m\},\ (p,q) \in G^t,\ (i,j) \in A \qquad \text{(Swap 1)}$$

$$\sum_{q \notin Q^t} x^t_{q,i,j} = \sum_{p \notin Q^t} x^t_{p,j,i} \qquad \forall t \in T \setminus \{m\},\ \forall (i,j) \in A \qquad \text{(Swap 2)}$$

$$\sum_{(i,j) \in A} y^t_{(p,q),(i,j)} = 1 \qquad \forall t \in T,\ \forall (p,q) \in G^t \qquad \text{(Gate 1)}$$

$$\{w^t_{p,i}, w^t_{q,j}\} \geq y^t_{(p,q),(i,j)} \geq \{0, w^t_{p,i} + w^t_{q,j} - 1\} \qquad \forall t \in T,\ \forall (p,q) \in G^t,\ \forall (i,j) \in A \qquad \text{(Gate 2)}$$

$$\sum_{(i,j) \in A} x^t_{q,i,j} \leq z^t \qquad \forall q \in Q,\ \forall t \in T^{\text{SWAP}} \qquad \text{(Count)}$$

Binary IP Formulation:

$$\min \sum_{t \in T^{\text{SWAP}}} z^t \qquad \leftarrow \; [z^t = 1 \quad \text{if swap layer } t \text{ used}]$$

$$\text{s.t.} \quad \sum_{j \in V} w^t_{q,j} = 1 \qquad \forall q \in Q,\ \forall t \in T \qquad \text{(Loc 1)}$$

$$\sum_{q \in Q} w^t_{q,j} = 1 \qquad \forall j \in V,\ \forall t \in T \qquad \text{(Loc 2)}$$

$$w^t_{q,i} = x^t_{q,i,i} + \sum_{k:(i,k) \in A} x^t_{q,i,k} \qquad \forall q \in Q,\ i \in V,\ \forall t \in T \setminus \{m\}, \qquad \text{(Route 1)}$$

$$w^t_{q,i} = x^{t-1}_{q,i,i} + \sum_{k:(k,i) \in A} x^{t-1}_{q,k,i} \qquad \forall q \in Q,\ \forall i \in V,\ \forall t \in T \setminus \{1\} \qquad \text{(Route 2)}$$

$$x^t_{p,i,j} = x^t_{q,j,i} \qquad \forall t \in T \setminus \{m\},\ (p,q) \in G^t,\ (i,j) \in A \qquad \text{(Swap 1)}$$

$$\sum_{q \notin Q^t} x^t_{q,i,j} = \sum_{p \notin Q^t} x^t_{p,j,i} \qquad \forall t \in T \setminus \{m\},\ \forall (i,j) \in A \qquad \text{(Swap 2)}$$

$$\sum_{(i,j) \in A} y^t_{(p,q),(i,j)} = 1 \qquad \forall t \in T,\ \forall (p,q) \in G^t \qquad \text{(Gate 1)}$$

$$\{w^t_{p,i}, w^t_{q,j}\} \geq y^t_{(p,q),(i,j)} \geq \{0, w^t_{p,i} + w^t_{q,j} - 1\} \qquad \forall t \in T,\ \forall (p,q) \in G^t,\ \forall (i,j) \in A \qquad \text{(Gate 2)}$$

$$\sum_{(i,j) \in A} x^t_{q,i,j} \leq z^t \qquad \forall q \in Q,\ \forall t \in T^{\text{SWAP}} \qquad \text{(Count)}$$

Binary IP Formulation:

$$\min \sum_{t \in T^{\text{SWAP}}} z^t \qquad \leftarrow \; [z^t = 1 \quad \text{if swap layer } t \text{ used}]$$

$$\text{s.t.} \sum_{j \in V} w^t_{q,j} = 1 \qquad \forall q \in Q,\ \forall t \in T \qquad \text{(Loc 1)}$$

$$\sum_{q \in Q} w^t_{q,j} = 1 \qquad \forall j \in V,\ \forall t \in T \qquad \text{(Loc 2)}$$

$$w^t_{q,i} = x^t_{q,i,i} + \sum_{k:(i,k) \in A} x^t_{q,i,k} \qquad \forall q \in Q,\ i \in V,\ \forall t \in T \setminus \{m\}, \qquad \text{(Route 1)}$$

$$w^t_{q,i} = x^{t-1}_{q,i,i} + \sum_{k:(k,i) \in A} x^{t-1}_{q,k,i} \qquad \forall q \in Q,\ \forall i \in V,\ \forall t \in T \setminus \{1\} \qquad \text{(Route 2)}$$

$$x^t_{p,i,j} = x^t_{q,j,i} \qquad \forall t \in T \setminus \{m\},\ (p,q) \in G^t,\ (i,j) \in A \qquad \text{(Swap 1)}$$

$$\sum_{q \notin Q^t} x^t_{q,i,j} = \sum_{p \notin Q^t} x^t_{p,j,i} \qquad \forall t \in T \setminus \{m\},\ \forall (i,j) \in A \qquad \text{(Swap 2)}$$

$$\sum_{(i,j) \in A} y^t_{(p,q),(i,j)} = 1 \qquad \forall t \in T,\ \forall (p,q) \in G^t \qquad \text{(Gate 1)}$$

$$\{w^t_{p,i}, w^t_{q,j}\} \geq y^t_{(p,q),(i,j)} \geq \{0, w^t_{p,i} + w^t_{q,j} - 1\} \qquad \forall t \in T,\ \forall (p,q) \in G^t,\ \forall (i,j) \in A \qquad \text{(Gate 2)}$$

$$\sum_{(i,j) \in A} x^t_{q,i,j} \leq z^t \qquad \forall q \in Q,\ \forall t \in T^{\text{SWAP}} \qquad \text{(Count)}$$

Binary IP Formulation:

$$\min \sum_{t \in T^{\text{SWAP}}} z^t \qquad \leftarrow \; [z^t = 1 \quad \text{if swap layer } t \text{ used}]$$

$$\text{s.t.} \sum_{j \in V} w^t_{q,j} = 1 \qquad \forall q \in Q,\ \forall t \in T \qquad \text{(Loc 1)}$$

$$\sum_{q \in Q} w^t_{q,j} = 1 \qquad \forall j \in V,\ \forall t \in T \qquad \text{(Loc 2)}$$

$$w^t_{q,i} = x^t_{q,i,i} + \sum_{k:(i,k) \in A} x^t_{q,i,k} \qquad \forall q \in Q,\ i \in V,\ \forall t \in T \setminus \{m\}, \qquad \text{(Route 1)}$$

$$w^t_{q,i} = x^{t-1}_{q,i,i} + \sum_{k:(k,i) \in A} x^{t-1}_{q,k,i} \qquad \forall q \in Q,\ \forall i \in V,\ \forall t \in T \setminus \{1\} \qquad \text{(Route 2)}$$

$$x^t_{p,i,j} = x^t_{q,j,i} \qquad \forall t \in T \setminus \{m\},\ (p,q) \in G^t,\ (i,j) \in A \qquad \text{(Swap 1)}$$

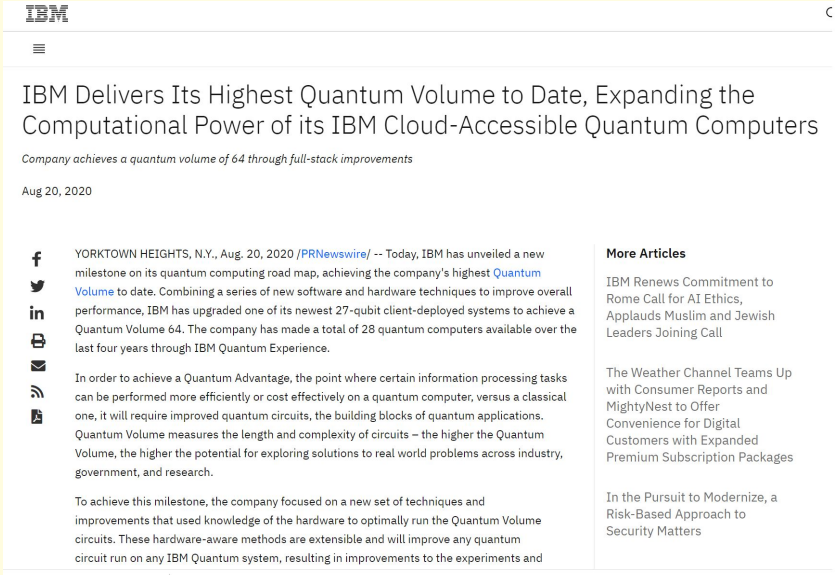
$$\sum_{q \notin Q^t} x^t_{q,i,j} = \sum_{p \notin Q^t} x^t_{p,j,i} \qquad \forall t \in T \setminus \{m\},\ \forall (i,j) \in A \qquad \text{(Swap 2)}$$

$$\sum_{(i,j) \in A} y^t_{(p,q),(i,j)} = 1 \qquad \forall t \in T,\ \forall (p,q) \in G^t \qquad \text{(Gate 1)}$$

$$\{w^t_{p,i}, w^t_{q,j}\} \geq y^t_{(p,q),(i,j)} \geq \{0, w^t_{p,i} + w^t_{q,j} - 1\} \qquad \forall t \in T,\ \forall (p,q) \in G^t,\ \forall (i,j) \in A \qquad \text{(Gate 2)}$$

$$\sum_{(i,j) \in A} x^t_{q,i,j} \leq z^t \qquad \forall q \in Q,\ \forall t \in T^{\text{SWAP}} \qquad \text{(Count)}$$

Quantum Volume 64



IBM

IBM Delivers Its Highest Quantum Volume to Date, Expanding the Computational Power of its IBM Cloud-Accessible Quantum Computers

Company achieves a quantum volume of 64 through full-stack improvements

Aug 20, 2020

YORKTOWN HEIGHTS, N.Y., Aug. 20, 2020 /PRNewswire/ -- Today, IBM has unveiled a new milestone on its quantum computing road map, achieving the company's highest Quantum Volume to date. Combining a series of new software and hardware techniques to improve overall performance, IBM has upgraded one of its newest 27-qubit client-deployed systems to achieve a Quantum Volume 64. The company has made a total of 28 quantum computers available over the last four years through IBM Quantum Experience.

In order to achieve a Quantum Advantage, the point where certain information processing tasks can be performed more efficiently or cost effectively on a quantum computer, versus a classical one, it will require improved quantum circuits, the building blocks of quantum applications. Quantum Volume measures the length and complexity of circuits – the higher the Quantum Volume, the higher the potential for exploring solutions to real world problems across industry, government, and research.

To achieve this milestone, the company focused on a new set of techniques and improvements that used knowledge of the hardware to optimally run the Quantum Volume circuits. These hardware-aware methods are extensible and will improve any quantum circuit run on any IBM Quantum system, resulting in improvements to the experiments and

More Articles

IBM Renews Commitment to Rome Call for AI Ethics, Applauds Muslim and Jewish Leaders Joining Call

The Weather Channel Teams Up with Consumer Reports and MightyNest to Offer Convenience for Digital Customers with Expanded Premium Subscription Packages

In the Pursuit to Modernize, a Risk-Based Approach to Security Matters

A second look at the formulation:

$$\min \sum_{t \in T^{\text{SWAP}}} z^t \qquad \leftarrow \text{ one possible objective}$$

$$\text{s.t.} \quad \sum_{j \in V} w^t_{q,j} = 1 \qquad \forall q \in Q,\ \forall t \in T \qquad \text{(Loc 1)}$$

$$\sum_{q \in Q} w^t_{q,j} = 1 \qquad \forall j \in V,\ \forall t \in T \qquad \text{(Loc 2)}$$

$$w^t_{q,i} = x^t_{q,i,i} + \sum_{k:(i,k) \in A} x^t_{q,i,k} \qquad \forall q \in Q,\ i \in V,\ \forall t \in T \setminus \{m\}, \qquad \text{(Route 1)}$$

$$w^t_{q,i} = x^{t-1}_{q,i,i} + \sum_{k:(k,i) \in A} x^{t-1}_{q,k,i} \qquad \forall q \in Q,\ \forall i \in V,\ \forall t \in T \setminus \{1\} \qquad \text{(Route 2)}$$

$$x^t_{p,i,j} = x^t_{q,j,i} \qquad \forall t \in T \setminus \{m\},\ (p,q) \in G^t,\ (i,j) \in A \qquad \text{(Swap 1)}$$

$$\sum_{q \notin Q^t} x^t_{q,i,j} = \sum_{p \notin Q^t} x^t_{p,j,i} \qquad \forall t \in T \setminus \{m\},\ \forall (i,j) \in A \qquad \text{(Swap 2)}$$

$$\sum_{(i,j) \in A} y^t_{(p,q),(i,j)} = 1 \qquad \forall t \in T,\ \forall (p,q) \in G^t \qquad \text{(Gate 1)}$$

$$\{w^t_{p,i}, w^t_{q,j}\} \geq y^t_{(p,q),(i,j)} \geq \{0, w^t_{p,i} + w^t_{q,j} - 1\} \qquad \forall t \in T,\ \forall (p,q) \in G^t,\ \forall (i,j) \in A \qquad \text{(Gate 2)}$$

$$\sum_{(i,j) \in A} x^t_{q,i,j} \leq z^t \qquad \forall q \in Q,\ \forall t \in T^{\text{SWAP}} \qquad \text{(Count)}$$

Theorem

Replacing the gate constraints

$$\sum_{(i,j)\in A} y^t_{(p,q),(i,j)} \;=\; 1 \qquad\qquad \forall t \in T,\ \forall (p,q) \in G^t$$

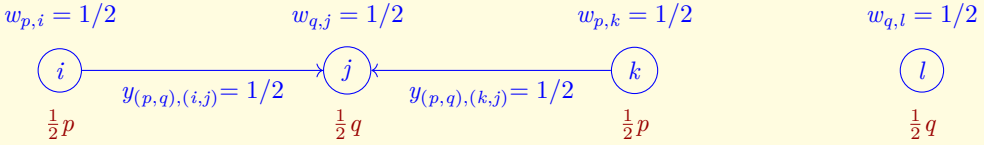
$$\{w^t_{p,i}, w^t_{q,j}\} \;\geq\; y^t_{(p,q),(i,j)} \;\geq\; \{0, w^t_{p,i} + w^t_{q,j} - 1\} \qquad \forall t \in T,\ \forall (p,q) \in G^t,\ \forall (i,j) \in A$$

with the following

$$w^t_{q,j} \;\leq\; \sum_{i:(i,j)\in A} w^t_{p,i} \qquad \forall t \in T,\ \forall (p,q) \in G^t,\ \forall j \in V$$

$$w^t_{p,i} \;\leq\; \sum_{j:(i,j)\in A} w^t_{q,j} \qquad \forall t \in T,\ \forall (p,q) \in G^t,\ \forall i \in V$$

gives a strictly stronger LP relaxation (after projecting out the y variables).



Theorem

Replacing the gate constraints

$$\sum_{(i,j)\in A} y^t_{(p,q),(i,j)} = 1 \qquad \forall t \in T,\ \forall (p,q) \in G^t$$

$$\{w^t_{p,i}, w^t_{q,j}\} \geq y^t_{(p,q),(i,j)} \geq \{0, w^t_{p,i} + w^t_{q,j} - 1\} \qquad \forall t \in T,\ \forall (p,q) \in G^t,\ \forall (i,j) \in A$$

with the following

$$w^t_{q,j} \leq \sum_{i:(i,j)\in A} w^t_{p,i} \qquad \forall t \in T,\ \forall (p,q) \in G^t,\ \forall j \in V$$

$$w^t_{p,i} \leq \sum_{j:(i,j)\in A} w^t_{q,j} \qquad \forall t \in T,\ \forall (p,q) \in G^t,\ \forall i \in V$$

gives a strictly stronger LP relaxation (after projecting out the y variables).

Note that for each $(p,q) \in \cup_{t\in T} G^t$,

- IP1 has $4|A| + 1$ constraints and $|A|$ auxiliary variables, (which dominates the size of IP1).
- Whereas, IP2 has $2|V|$ constraints for each $(p,q) \in \cup_{t\in T} G^t$.

A different objective function: error rate

- The quantum circuit is executed correctly if all gates and swaps are executed without error.
- For each edge (i, j) of the hardware graph
  - A gate (p, q) executes successfully with probability $P(p, q, i, j)$
  - A gate (p, q) executes and swaps qubits successfully with probability $P_{\mathsf{SWAP}}(p, q, i, j)$
  - A (non-gate) qubit q swaps successfully with probability $Q_{\mathsf{SWAP}}(q, i, j)$
- To maximize the success probability we can use the following linear function:

$$\max \sum_{t \in T} \sum_{(p,q) \in G^t} \sum_{(i,j) \in A} \log\left(P(p, q, i, j)\right) \left[y^t_{(p,q),(i,j)} - \frac{x^t_{p,i,j} + x^t_{q,j,i}}{2} \right]$$
$$+ \sum_{t \in T} \sum_{(p,q) \in G^t} \sum_{(i,j) \in A} \log\left(P_{\mathsf{SWAP}}(p, q, i, j)\right) \left[\frac{x^t_{p,i,j} + x^t_{q,j,i}}{2} \right]$$
$$+ \sum_{t \in T} \sum_{q \in Q \setminus Q^t} \sum_{(i,j) \in A} \log\left(Q_{\mathsf{SWAP}}(q, i, j)\right) x^t_{q,i,j}$$

Computational Experiments

Algorithm	HOP (line)	HOP (Y)
BIP	0.6771 ± 0.0022	0.6261 ± 0.0019
SABRE	0.6506 ± 0.0030	0.6062 ± 0.0020
BIP-Layout	0.6292 ± 0.0031	0.6000 ± 0.0022
BIP-Routing	0.6721 ± 0.0023	0.6221 ± 0.0018
BIP-Constrained	0.6710 ± 0.0023	0.6189 ± 0.0019

Table: Heavy output probability measured on a line topology and a Y topology

- BIP, minimizing error rate first, and depth as the second objective.
- Qiskit's default qubit allocation algorithm SABRE
- BIP for initial qubit assignment, then SABRE for routing.
- SABRE for initial qubit assignment, then BIP for routing
- The BIP constrained that the final qubit arrangement is same as the initial qubit assignment.

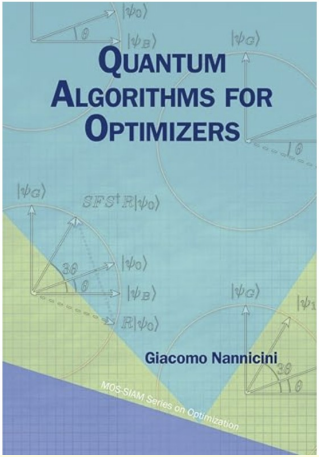
Computational Experiments

	Line		Y		Grid	
Algorithm	CNOT	Dur.	CNOT	Dur.	CNOT	Dur.
BIP	67.2	8.1	67.2	9.4	46.9	4.2
SABRE	73.8	9.7	71.4	10.9	54.0	6.4
BIP-Layout	75.8	10.4	72.1	11.1	55.6	6.7
BIP-Routing	70.5	8.8	69.9	10.0	49.1	4.8
BIP-Constrained	70.5	8.7	69.8	9.9	48.8	4.8

Table: Average CNOT count and duration (microseconds) for the 6-qubit quantum volume circuits.

- BIP, minimizing error rate first, and depth as the second objective.
- Qiskit's default qubit allocation algorithm SABRE
- BIP for initial qubit assignment, then SABRE for routing.
- SABRE for initial qubit assignment, then BIP for routing
- The BIP constrained that the final qubit arrangement is same as the initial qubit assignment.

if interested in quantum algorithms



Quantum Algorithms for Optimizers

by Giacomo Nannicini (Author) | Format: Paperback

This book presents a self-contained introduction to quantum algorithms, with a focus on quantum optimization —quantum approaches to solving optimization problems. It equips readers with the essential tools to assess the strengths and limitations of these algorithms, emphasizing provable guarantees and computational complexity. The first comprehensive treatment of quantum optimization, Quantum Algorithms for Optimizers provides a rigorous introduction to the computational model of quantum computers and to the theory of quantum algorithms, contains detailed discussions of some of the most important developments in quantum optimization algorithms, and summarizes the most significant advances in the open literature.

Report an issue with this product or seller

ISBN-10	ISBN-13	Publisher	Publication date	Language
1611978750	978-1611978759	SIAM - Society for Industrial and Applied...	December 17, 2025	English

Parallel Token Swapping

(Joint work with Ricky Shapley and Ishan Bansal)

Time periods in the formulation:

$$\min \sum_{t \in T^{\mathsf{SWAP}}} z^t \qquad \leftarrow \text{ one possible objective}$$

s.t.

$$\sum_{j \in V} w^t_{q,j} = 1 \qquad \forall q \in Q,\ t \in T \qquad \text{(Loc 1)}$$

$$\sum_{q \in Q} w^t_{q,j} = 1 \qquad \forall j \in V,\ t \in T \qquad \text{(Loc 2)}$$

$$w^t_{q,i} = x^t_{q,i,i} + \sum_{k:(i,k) \in A} x^t_{q,i,k} \qquad \forall q \in Q,\ i \in V,\ t \in T \setminus \{m\}, \qquad \text{(Route 1)}$$

$$w^t_{q,i} = x^{t-1}_{q,i,i} + \sum_{k:(k,i) \in A} x^{t-1}_{q,k,i} \qquad \forall q \in Q,\ i \in V,\ t \in T \setminus \{1\} \qquad \text{(Route 2)}$$

$$x^t_{p,i,j} = x^t_{q,j,i} \qquad \forall t \in T \setminus \{m\},\ (p,q) \in G^t,\ (i,j) \in A \qquad \text{(Swap 1)}$$

$$\sum_{q \notin Q^t} x^t_{q,i,j} = \sum_{p \notin Q^t} x^t_{p,j,i} \qquad \forall t \in T \setminus \{m\},\ (i,j) \in A \qquad \text{(Swap 2)}$$

$$w^t_{q,j} \le \sum_{i:(i,j) \in A} w^t_{p,i} \qquad \forall t \in T,\ (p,q) \in G^t,\ j \in V \qquad \text{(Gate 1}^*\text{)}$$

$$w^t_{p,i} \le \sum_{j:(i,j) \in A} w^t_{q,j} \qquad \forall t \in T,\ (p,q) \in G^t,\ i \in V \qquad \text{(Gate 2}^*\text{)}$$

$$\sum_{(i,j) \in A} x^t_{q,i,j} \le z^t \qquad \forall q \in Q,\ t \in T^{\mathsf{SWAP}} \qquad \text{(Count)}$$

Why do we need SWAP layers?

The time periods in this formulation have two flavors:

$$T = T^{Active} \cup T^{SWAP}$$

where $|T^{SWAP}| \approx k\,|T^{Active}|$, and k is a heuristically chosen integer.

- T^{Active} is the time steps when useful action happens
- T^{SWAP} is needed to keep track of the time it takes rearrange the qubits.

$$T = \Big\{ \underbrace{T^{A1}}_{\text{gate group } G^1}, \ldots\ldots, \underbrace{T^{A5}}_{\text{gate group } G_5}, \underbrace{T^{S51}, T^{S52}, T^{S53}, T^{S54}, T^{S55}}_{\textit{SWAP layers}}, \underbrace{T^{A6}}_{\text{gate group } G^6}, \ldots\ldots \Big\}$$

Question: Can we design a function

$$f(w^S, w^T)$$

to estimate the (shortest) time it would take to move configuration w^S to w^T?

Why do we need SWAP layers?

The time periods in this formulation have two flavors:

$$T \;=\; T^{Active} \cup T^{SWAP}$$

where $|T^{SWAP}| \approx k\,|T^{Active}|$, and k is a heuristically chosen integer.

- T^{Active} is the time steps when useful action happens
- T^{SWAP} is needed to keep track of the time it takes rearrange the qubits.

$$T = \Big\{ \underbrace{T^{A1}}_{\text{gate group } G^1}, \ldots\ldots, \underbrace{T^{A5}}_{\text{gate group } G_5}, \underbrace{T^{S51}, T^{S52}, T^{S53}, T^{S54}, T^{S55}}_{\textit{SWAP layers}}, \underbrace{T^{A6}}_{\text{gate group } G^6}, \ldots\ldots \Big\}$$

Question: Can we design a function

$$f(w^S, w^T)$$

to estimate the (shortest) time it would take to move configuration w^S to w^T?

A possible better formulation for Qubit routing

- If we can construct this function f, we can work with a much simpler formulation:
  - No need for all the swap layers T^{SWAP}
  - No need for all the routing/swap variables and constraints.

$$\min \quad \sum_{t=1}^{|T^{Active}|-1} f(w^t, w^{t+1})$$

$$\text{s.t.} \quad \sum_{j \in V} w^t_{q,j} = 1 \qquad \forall t \in T^{Active}, \ \forall q \in Q, \qquad \text{(Loc 1)}$$

$$\sum_{q \in Q} w^t_{q,j} = 1 \qquad \forall t \in T^{Active}, \ \forall j \in V \qquad \text{(Loc 2)}$$

$$w^t_{q,j} \leq \sum_{i:(i,j) \in A} w^t_{p,i} \qquad \forall t \in T^{Active}, \ \forall (p,q) \in G^t, \ \forall j \in V \qquad \text{(Gate 1*)}$$

$$w^t_{p,i} \leq \sum_{j:(i,j) \in A} w^t_{q,j} \qquad \forall t \in T^{Active}, \ \forall (p,q) \in G^t, \ \forall i \in V \qquad \text{(Gate 2*)}$$

Computing/estimating reconfiguration time

- Given the location of all qubits at step t and $t+1$ in the hardware graph G: (i.e. w^t and w^{t+1})
  (Remember $w^t_{q,i} = 1$ means that at time step t qubit q is located at node i)
- Find a function to compute/estimate how long it will take to reconfigure them: $f(w^t, w^{t+1})$
- If possible, do it using linear inequalities.

Max-distance lower bound

Let $\mathrm{dist}(q)$ denote the distance in G between the location of qubit q at time t and $t+1$.

The *max-distance* bound is

$$\mathrm{LB}_{\max} = \max_{q \in Q}\{\mathrm{dist}(q)\}.$$

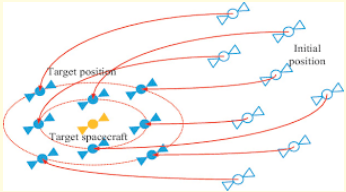
Clearly any reconfiguration needs at least $\mathrm{LB}_{\max}$ SWAP steps.

Therefore

$$f(w^t, w^{t+1}) \geq \mathrm{LB}_{\max}(G, w^t, w^{t+1})$$

Combinatorial configuration problems

- Given an initial configuration of objects (tokens, counters, agents, ...)
- Move them to a final configuration following a specified rule set.



Relevant questions:

- Is there a feasible routing?
- If so, what is the most efficient way?
- Can we compute upper and lower bounds on OPT?

Parallel Token Swapping Problem

Graph: $G = (V, E)$.

Tokens: duplicate of the vertex set, $\mathcal{T} \cong V$.

- We use the identification $\mathcal{T} = \{t_v : v \in V\}$, a copy of V.

Configuration: a bijection $\pi : V \to \mathcal{T}$.

- $\pi(v)$ is the token currently on node v.
- Final (target) configuration w.l.o.g. is the identity $\mathrm{Id} : v \mapsto t_v$.

Parallel Token Swapping (PTS)

Input: A graph $G = (V, E)$ and an initial configuration π.

Question: Find a sequence of matchings $(M_1, \ldots, M_k)$ of *minimum* length k such that

$$M_k \cdots M_1 \pi \;=\; \mathrm{Id}.$$

[here a matching means a symmetric permutation matrix where nonzeroes agree with edges]

Token swapping literature:

Non-parallel token swapping:

- Rearrangement of tokens is always possible in at most $O(n^2)$ swaps. [Yamanaka et al. 2015]
- The problem is NP-HARD but 4-approximation is possible [Miltzow et al. 2016]
- NP-HARD even on graphs with constant treewidth [Bonnet 2018].
- Polynomial time exact algorithms are known for cliques [Cayley 1849], line graphs [knut 1973], cycles [Jerrum 1985], star graphs [Portier 1990], complete bipartite graphs [Yamanaka2015], ...

Parallel token swapping:

- Any instance on an n-vertex graph can be solved in $3n$ steps [Alon 1994]
- Later improved to $3n/2 + O(\log n)$ [Zhang 1999]
- NP-HARD to decide if the optimal solution value is equal to three [kawahara 2017]
- NP-HARD on subdivided stars [Aicholzer et al. 2022]
- Not much else is know: Even the basic question

  "Is the problem NP-HARD on line graphs?"

  has been open for over three decades. [There is an OPT+1 algorithm]

Token swapping literature:

Non-parallel token swapping:

- Rearrangement of tokens is always possible in at most $O(n^2)$ swaps. [Yamanaka et al. 2015]
- The problem is NP-HARD but 4-approximation is possible [Miltzow et al. 2016]
- NP-HARD even on graphs with constant treewidth [Bonnet 2018].
- Polynomial time exact algorithms are known for cliques [Cayley 1849], line graphs [knut 1973], cycles [Jerrum 1985], star graphs [Portier 1990], complete bipartite graphs [Yamanaka2015], ...

Parallel token swapping:

- Any instance on an n-vertex graph can be solved in $3n$ steps [Alon 1994]
- Later improved to $3n/2 + O(\log n)$ [Zhang 1999]
- NP-HARD to decide if the optimal solution value is equal to three [kawahara 2017]
- NP-HARD on subdivided stars [Aicholzer et al. 2022]
- Not much else is know: Even the basic question

  "Is the problem NP-HARD on line graphs?"

  has been open for over three decades. [There is an OPT+1 algorithm]

Lower bounds and the stretch factor

Lower bound (LB): a function $\ell(G, \pi)$ such that

$$\mathrm{OPT}(G, \pi) \;\geq\; \ell(G, \pi) \quad \text{for all instances.}$$

Stretch factor of LB ℓ: Worst case performance of the bound considering all graphs and initial configurations.

$$\sigma(\ell) \;=\; \max_{(G,\pi)} \frac{\mathrm{OPT}(G, \pi)}{\ell(G, \pi)}.$$

Max-distance LB. The bound we saw earlier

$$\mathrm{LB}_{\max}(G, \pi) = \max_{q \in Q} \big\{ \mathrm{dist}(q) \big\}.$$

Question: How good is this lower bounds (in terms of stretch factor)?

Lower bounds and the stretch factor

Lower bound (LB): a function $\ell(G, \pi)$ such that

$$\mathrm{OPT}(G, \pi) \;\geq\; \ell(G, \pi) \quad \text{for all instances.}$$

Stretch factor of LB ℓ: Worst case performance of the bound considering all graphs and initial configurations.

$$\sigma(\ell) \;=\; \max_{(G,\pi)} \frac{\mathrm{OPT}(G, \pi)}{\ell(G, \pi)}.$$

Max-distance LB. The bound we saw earlier

$$\mathrm{LB}_{\max}(G, \pi) = \max_{q \in Q} \big\{ \mathrm{dist}(q) \big\}.$$

Question: How good is this lower bounds (in terms of stretch factor)?

Theorem

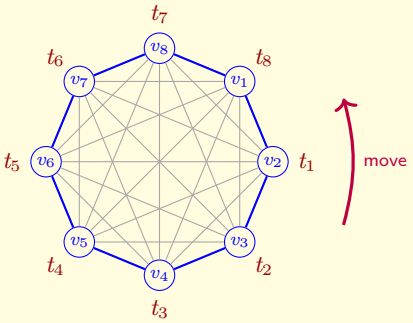
If $\ell(G, \pi)$ is a function of $\{\mathrm{dist}(1), \mathrm{dist}(2), \ldots, \mathrm{dist}(n)\}$ only, then

$$\max_{(G,\pi)} \frac{\mathrm{OPT}(G, \pi)}{\ell(G, \pi)} = \Omega(n)$$

i.e., the stretch of $\ell(G, \pi)$ can grow linearly with the number of vertices.

Proof.

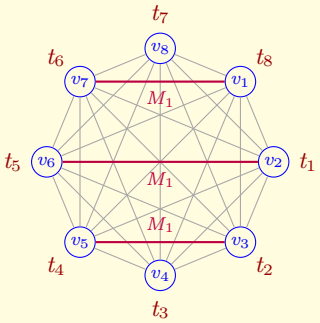
- We will generate two instances on
  1. A cycle graph, and
  2. A complete graph
- Tokens want to move counter-clockwise
  [$\mathrm{dist}(t) = 1$ for all tokens t]
- We will show that
  1. $\mathrm{OPT}(G, \pi) = 2$ for the complete graph, and
  2. $\mathrm{OPT}(G, \pi) = n - 1$ for the cycle graph



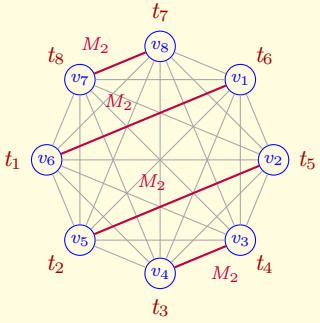
Complete graph

Proof.

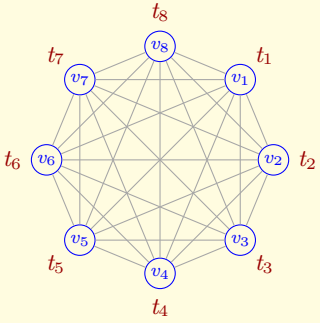
Tokens can be routed counter-clockwise with 2 matchings (parallel swaps):



π



$M_1\pi$



$M_2M_1\pi$

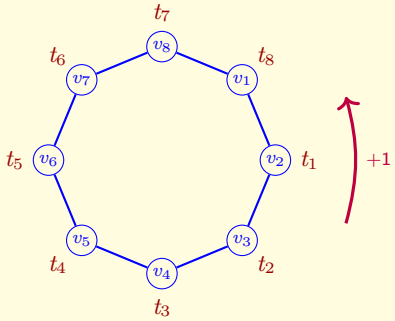
$$\implies \quad OPT = 2 \quad \implies \ell(G, \pi) \leq 2$$

Cycle graph

Proof.

- Each token starts with a charge of 0,
  - increase by 1 if moves counter-clockwise,
  - decrease by one if it moves clockwise
- Total charge of tokens at the beginning is zero
- Swaps do not change total charge
- Total charge of tokens at the end is zero
- At the end all tokens must have charge $+1 \pmod{n}$
- At the end at least one token must have negative charge
- That charge is $-n+1$ or $-2n+1$ or $-3n+1$ or ...
- Need at least $n-1$ matchings $\implies$ $OPT \geq n-1$
  [and this can be achieved]

  $\implies$ stretch factor is at least $(n-1)/2$.



The stretch factor of $d_{\max}$

- Therefore if $\ell(G, \pi)$ is a function of $\{\mathrm{dist}(1), \mathrm{dist}(2), \ldots, \mathrm{dist}(n)\}$ only, then the stretch factor of $\ell(G, \pi)$ can grow linearly with the number of vertices.
- One has to use information about the instance beyond distances.
- Topology of the graph and the initial configuration must be taken into account simultaneously.
- This motivates studying different families of graphs separately. For example:
  - Line graphs, Cycle graphs, Subdivided stars, Grid graphs, ...

Theorem

(a) *The stretch factor of $d_{\max}$ for line graphs is 2.*

(b) *The stretch factor of $d_{\max}$ for n-cycle graphs is between n and $n-1$.*

(c) *The stretch factor of $d_{\max}$ for subdivided star graphs is $\Theta(h)$ where h is the maximum degree of a vertex in the graph.*

The stretch factor of $d_{\max}$

- Therefore if $\ell(G, \pi)$ is a function of $\{\mathrm{dist}(1), \mathrm{dist}(2), \ldots, \mathrm{dist}(n)\}$ only, then the stretch factor of $\ell(G, \pi)$ can grow linearly with the number of vertices.
- One has to use information about the instance beyond distances.
- Topology of the graph and the initial configuration must be taken into account simultaneously.
- This motivates studying different families of graphs separately. For example:
  - Line graphs, Cycle graphs, Subdivided stars, Grid graphs, ...

Theorem

(a) *The stretch factor of $d_{\max}$ for line graphs is 2.*

(b) *The stretch factor of $d_{\max}$ for n-cycle graphs is between n and $n-1$.*

(c) *The stretch factor of $d_{\max}$ for subdivided star graphs is $\Theta(h)$ where h is the maximum degree of a vertex in the graph.*

Stretch factor of $d_{\max}$ on a line

Theorem

The stretch factor of d_{max} for line graphs is 2.

- The proof has three main steps.
  1. First, define a (non-negative) potential function for each token.

$$\Phi_\pi(t) = \max_{Q^- \in Q_\pi^-(t)} \left(\phi_\pi^-(t, Q^-) + \psi_\pi^-(t, \overline{Q^-})\right) + \max_{Q^+ \in Q_\pi^+(t)} \left(\phi_\pi^+(t, Q^+) + \psi_\pi^+(t, \overline{Q^+})\right).$$

  2. Then show that each iteration of the odd-even algorithm decreases the potential value by at least one for each token with positive (nonzero) potential.
  3. Finally argue that the potential of any token is bounded by $2d_{\max} - 1$.

Theorem (Kawahara et al. 2017)

On a line graph with n nodes, one can find a solution with at most $OPT + 1$ in poly-time.

Open problem

Is this problem poly-time solvable or is it NP-hard?

Stretch factor of $d_{\max}$ on a line

Theorem

The stretch factor of d_{max} for line graphs is 2.

- The proof has three main steps.
  1. First, define a (non-negative) potential function for each token.

$$\Phi_\pi(t) = \max_{Q^- \in Q_\pi^-(t)} \left(\phi_\pi^-(t, Q^-) + \psi_\pi^-(t, \overline{Q^-})\right) + \max_{Q^+ \in Q_\pi^+(t)} \left(\phi_\pi^+(t, Q^+) + \psi_\pi^+(t, \overline{Q^+})\right).$$

  2. Then show that each iteration of the odd-even algorithm decreases the potential value by at least one for each token with positive (nonzero) potential.
  3. Finally argue that the potential of any token is bounded by $2d_{\max} - 1$.

Theorem (Kawahara et al. 2017)

On a line graph with n nodes, one can find a solution with at most $OPT + 1$ in poly-time.

Open problem

Is this problem poly-time solvable or is it NP-hard?

PTS on a line (parallel sorting)

Theorem (Knuth, 1973)

The PTS problem can be solved on a line graph with n nodes in at most n steps.

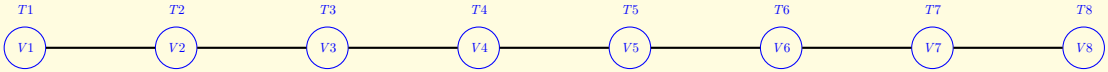
Algorithm Odd-Even Algorithm

Input: A configuration π on a line graph with n nodes and $n-1$ edges

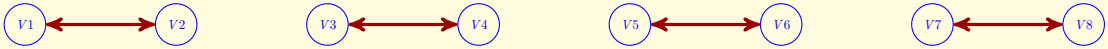
$E^O \leftarrow \{(i, i+1) : i \text{ is odd}\}$, $E^E \leftarrow \{(i, i+1) : i \text{ is even}\}$
$\pi_0 \leftarrow \pi$
$k \leftarrow 0$
while π_k is not Identity **do**
    $k \leftarrow k + 1$
    if k is odd **then**
        $M_k = \{(i, i+1) \in E^O : \pi_{k-1}(i) > \pi_{k-1}(i+1)\}$
    else
        $M_k = \{(i, i+1) \in E^E : \pi_{k-1}(i) > \pi_{k-1}(i+1)\}$
    $\pi_k \leftarrow M_k \pi_{k-1}$
return $M_1, \ldots, M_k$

Odd-Even (Parallel) Sorting

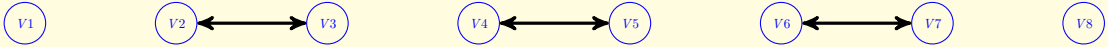
Given tokens on a line graph (or, an array):



Odd edges:



Even edges:



- Alternating between odd and even edges, flip the tokens if they want to cross each other

Odd-Even (Parallel) Sorting – example

- Consider n tokens arranged on a path.
- **Odd phase:** simultaneously compare-exchange tokens on odd edges
- **Even phase:** simultaneously compare-exchange tokens on even edges
- Repeat odd and even phases until the sequence is sorted.
- Example:
  - Initial sequence (configuration): $4\ 1\ 6\ 3\ 5\ 2$
  - Odd: $[4\ 1]\ [6\ 3]\ [5\ 2] \longrightarrow [1\ 4]\ [3\ 6]\ [2\ 5]$
  - Even: $1\ [4\ 3]\ [6\ 2]\ 5 \longrightarrow 1\ [3\ 4]\ [2\ 6]\ 5$
  - Odd: $[1\ 3]\ [4\ 2]\ [6\ 5] \longrightarrow [1\ 3]\ [2\ 4]\ [5\ 6]$
  - Even: $1\ [3\ 2]\ [4\ 5]\ 6 \longrightarrow 1\ [2\ 3]\ [4\ 5]\ 6 \rightarrow$ done.

Algorithmic results (based on the Odd-Even Algorithm)

Theorem

If G is an n-cycle, then,

(a) *If n is even, we can find a solution with value at most $2\,OPT$ in polynomial time.*

(b) *If n is odd, we can find a solution with value at most $2\,OPT + 1$ in polynomial time.*

Theorem

If G is a subdivided star with maximum degree h, then we can find a solution with value at most $4\,OPT + \min\{OPT, h\} + 1$ in polynomial time.

Theorem

If G is a $h \times n$ grid graph with $h \leq n$, then

(a) *If $h = 2$, we can find a solution with value at most $2\,OPT + 2$ in polynomial time.*

(b) *If $h \geq 3$, we can find a solution with value at most $2\,OPT + 2h$ in polynomial time.*

A variant: Colored PTS

Consider the following variant of the parallel token swapping problem, where tokens have colors.

Colors: Let $L = \{1, 2, \ldots, \ell\}$ be color labels and $\mathcal{L} : \mathcal{T} \to L$ a token labeling.

Colored Parallel Token Swapping

Input: $G = (V, E)$, initial π, token colors $\mathcal{L}$.

Question: find matchings $(M_1, \ldots, M_k)$ of minimum k such that

$$\mathcal{L}\big((M_k \cdots M_1 \pi)(v)\big) \;=\; \mathcal{L}\big(\mathrm{Id}(v)\big) \quad \text{for all } v \in V.$$

(each node requires a token of the correct *color*, not necessarily a specific token.)

Summary of results

	PTS	d_{max} stretch	colored PTS
Line Graphs	$OPT + 1$ [Kawahara 2017]	$\mathbf{2}$	$OPT + 1$ [Kawahara 2017]
Even Cycles	$\mathbf{2\,OPT}$	$\mathbf{n}$	$\mathbf{2\,OPT}$
Odd Cycles	$\mathbf{2\,OPT + 1}$	$\mathbf{n}$	$\mathbf{2\,OPT + 1}$
Subdivided Stars	$\mathbf{4\,OPT} + \min\{\mathbf{OPT}, \mathbf{h}\} + \mathbf{1}$	$(\mathbf{h})$	$\mathbf{4\,OPT} + \min\{\mathbf{OPT}, \mathbf{h}\} + \mathbf{1}$
Ladder Graphs	$\mathbf{2\,OPT + 2}$	—	$\mathbf{2\,OPT + 2}$
Grid Graphs	$\mathbf{2\,OPT + 2h}$	—	$\mathbf{2\,OPT + 2h}$

Table: A boldface indicates our results. Previous best factors for were $O(n)$ [Alon 1994, Zhang 1999].

PTS Note: on subdivided stars is NP-hard, others (including line graphs are unknown)

thank you.