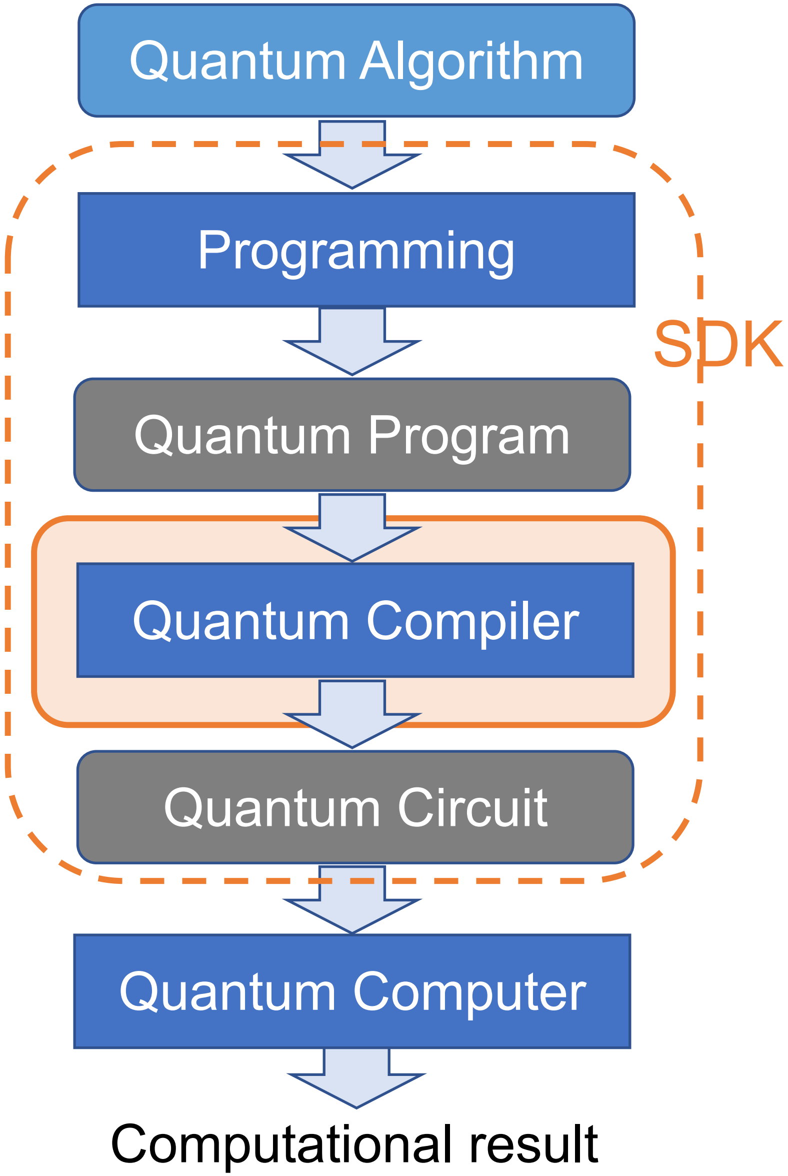
Quantum Circuit Compilers Using Gate Commutation Rules

Toshinari Itoko, Rudy Raymond, Takashi Imamichi, Atsushi Matsuo, Andrew W. Cross

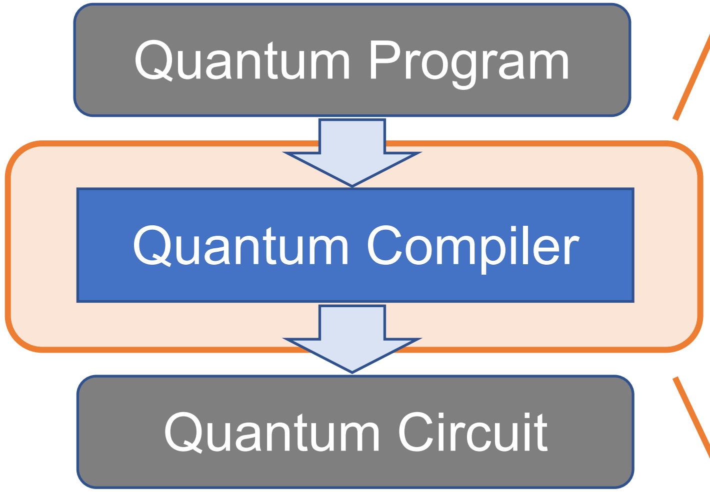
IBM Research

Quantum compiler



- Dozens of **quantum algorithms** have been developed (e.g. see "Quantum Algorithm Zoo").
  - Factoring [Shor 1994], Database search [Grover 1996]
- Recent advance in HW technologies has enabled anyone with access to real **quantum computers**.
  - IBM (2016), Rigetti Computing (2017), Alibaba (2018)
- To run quantum algorithms on quantum computer, we need Software Development Kit (SDK) including **quantum (circuit) compiler**.

Three major functionalities in quantum compiler



- Decomposition

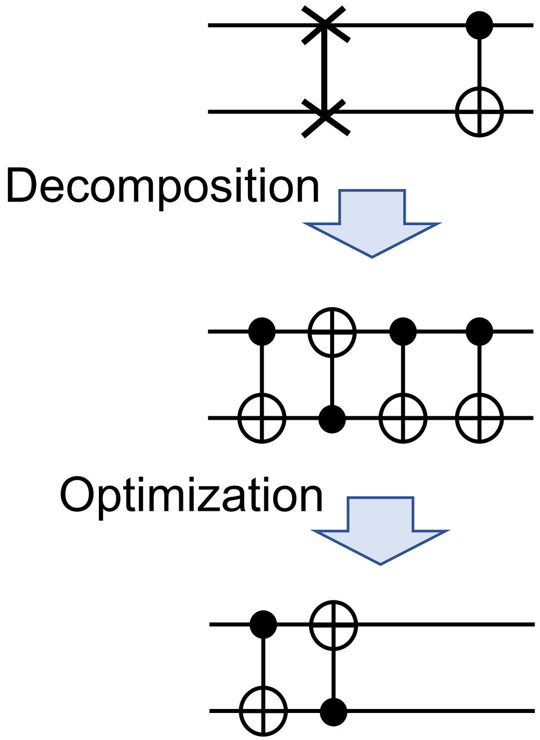
  Decompose a program into elementary operations (gates)

- Optimization

  Optimize a quantum circuit (sequence of gates)

- **Mapping**

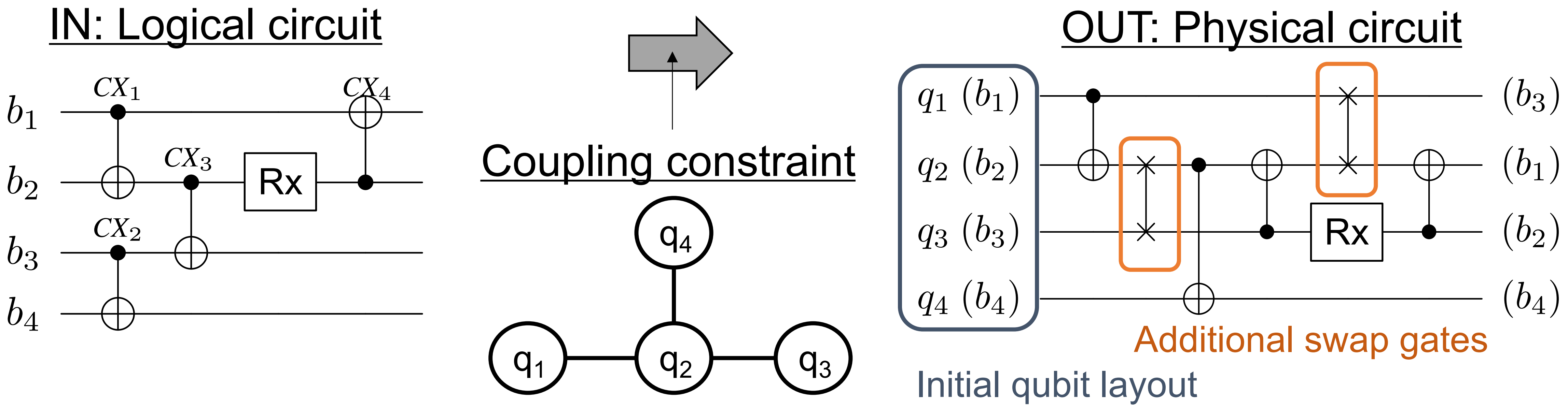
  Transform a logical quantum circuit into an physical one **satisfying processor-dependent constraints**



Swap gate mapping

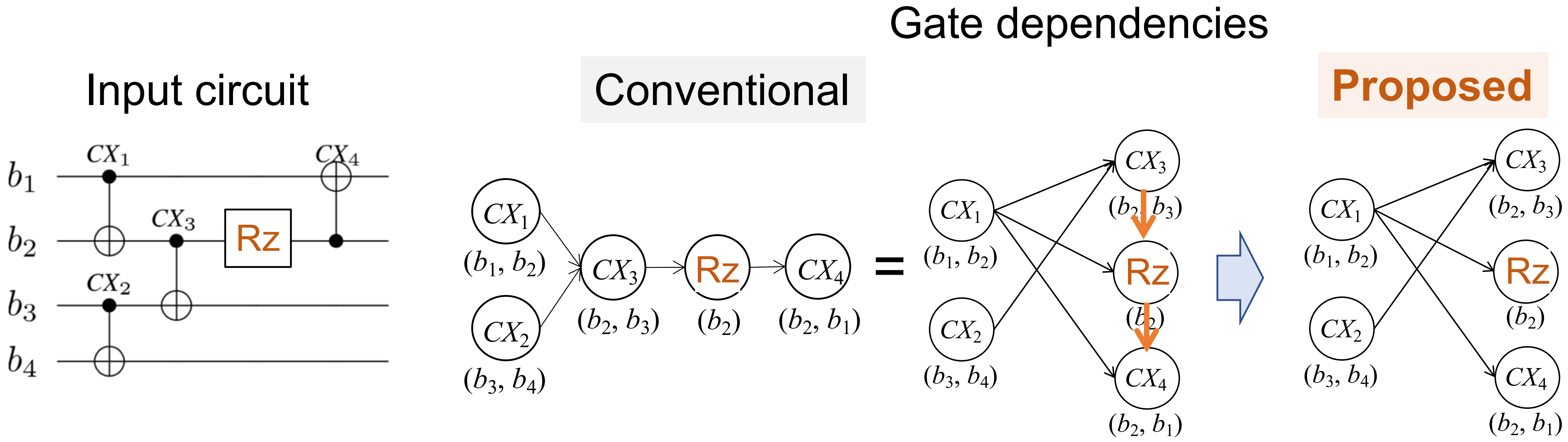
- Find a **mapping**: Logical circuit -> Physical circuit
  - Mapping = Initial qubit **layout** + Additional **swap gates**
  - Layout = Allocation of logical qubits to physical qubits
- Subject to **coupling constraint**

Minimize



Our research contribution

1. Formulation of Minimum Swap Gate Mapping (MSGM) with **considering gate commutation rules**
   - Decrease gate dependencies = increase search space
2. A **better heuristic algorithm** for solving MSGM



Quantum circuit

Quantum circuit is a sequence of elementary quantum gates.

We consider a **universal gate set** {Rx, Rz, CNOT}.

- One-qubit rotation gates: **Rx, Rz**

Rx[θ](b2)

Acting qubit

Rotation angle (omit in this talk)

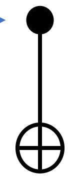
- Controlled-NOT gate (**CNOT**)

Control qubit

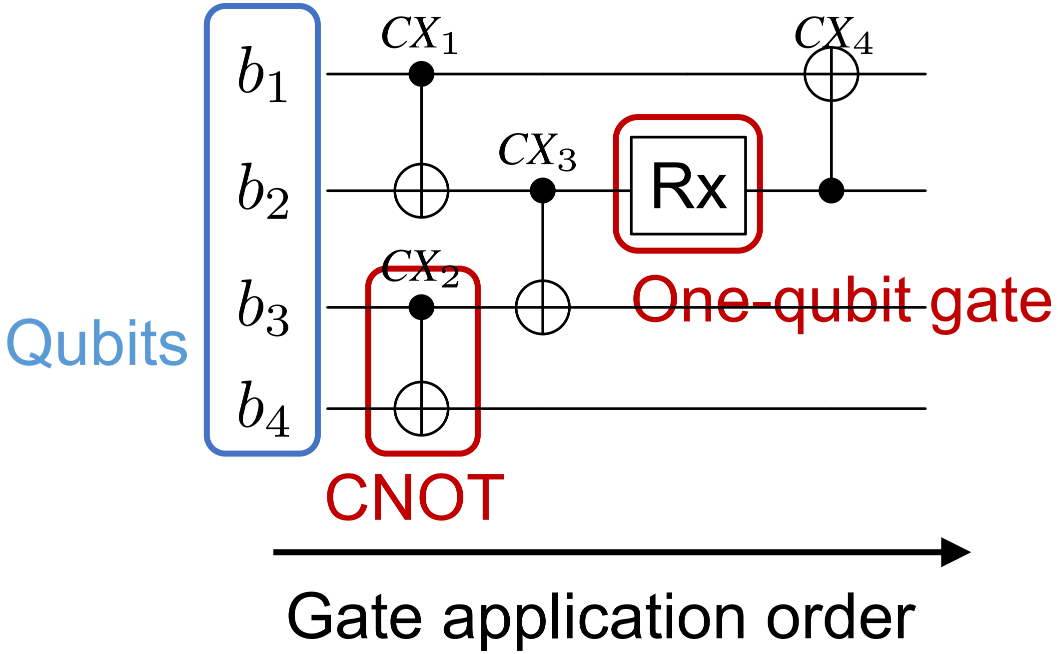
CNOT(b3, b4)

Acting qubits

Target qubit

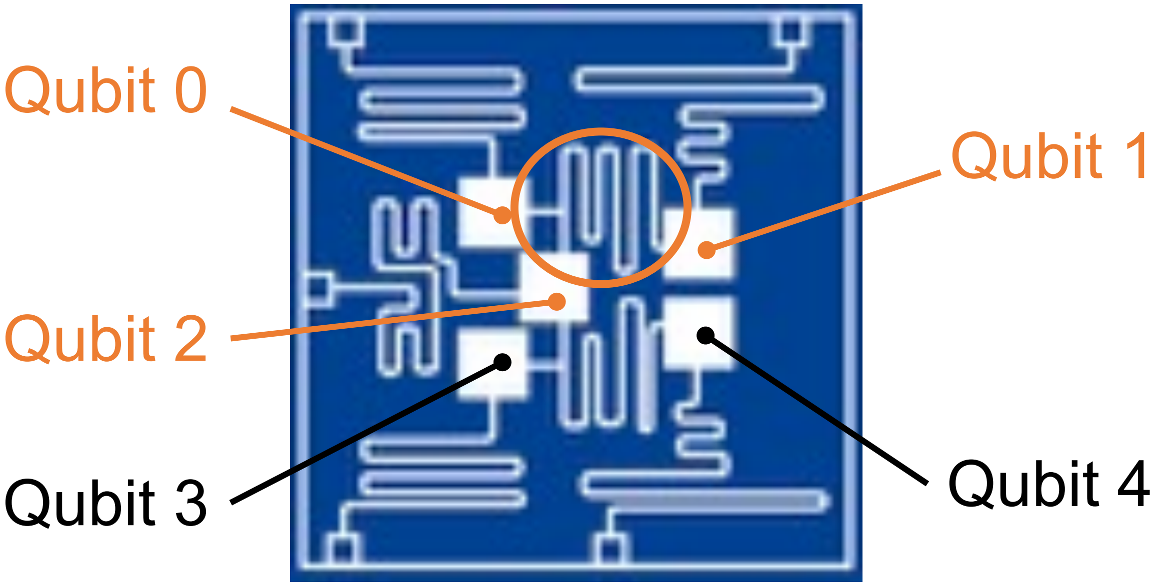


Ex) Quantum circuit



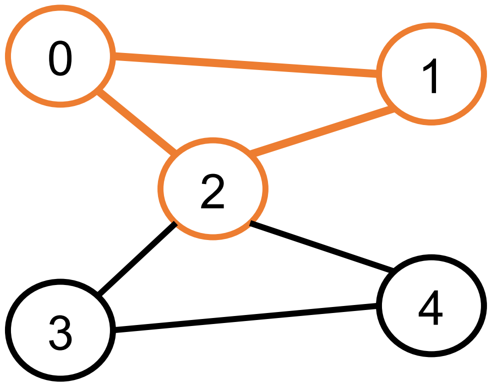
Coupling constraint

Only two-qubit (CNOT) gates on coupled qubits are allowed, which is represented by coupling graph.



Device image of IBM Q 5 Tenerife [ibmqx4]

Coupling Graph

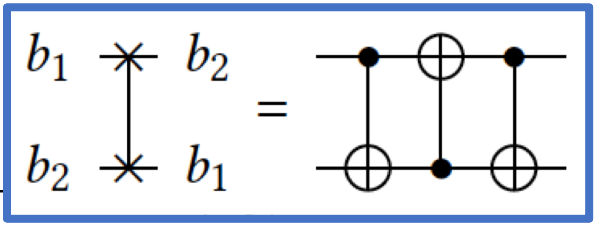


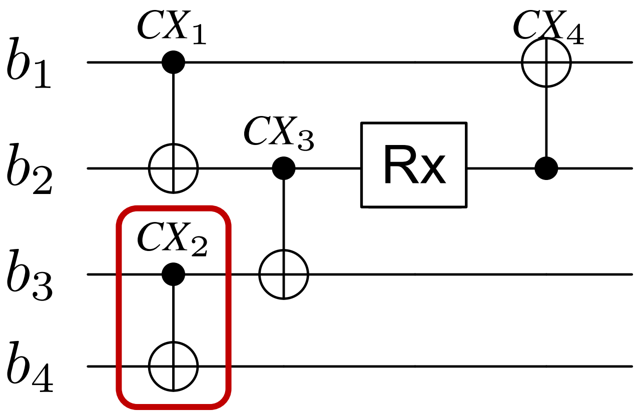
Node ⇔ Qubit
Edge ⇔ Coupling

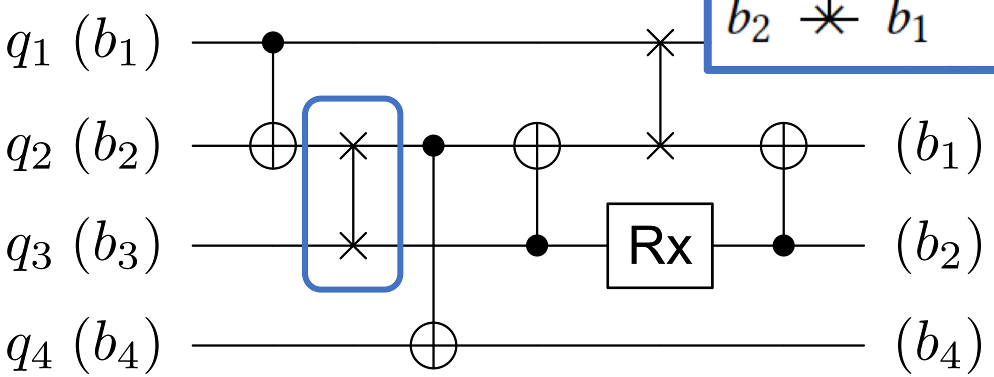
Constraint satisfaction by adding swap gate

Coupling constraint can be resolved by adding **swap gates**.

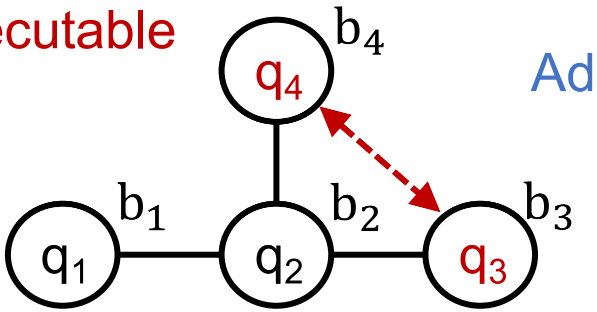
Example (Initial qubit layout: b_i -> q_i for i = 1,2,3,4)



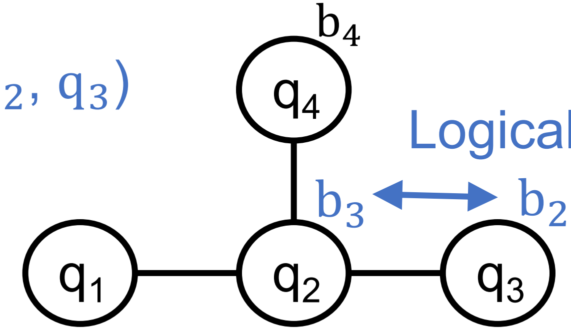




Not executable



Add SWAP(q_2, q_3)

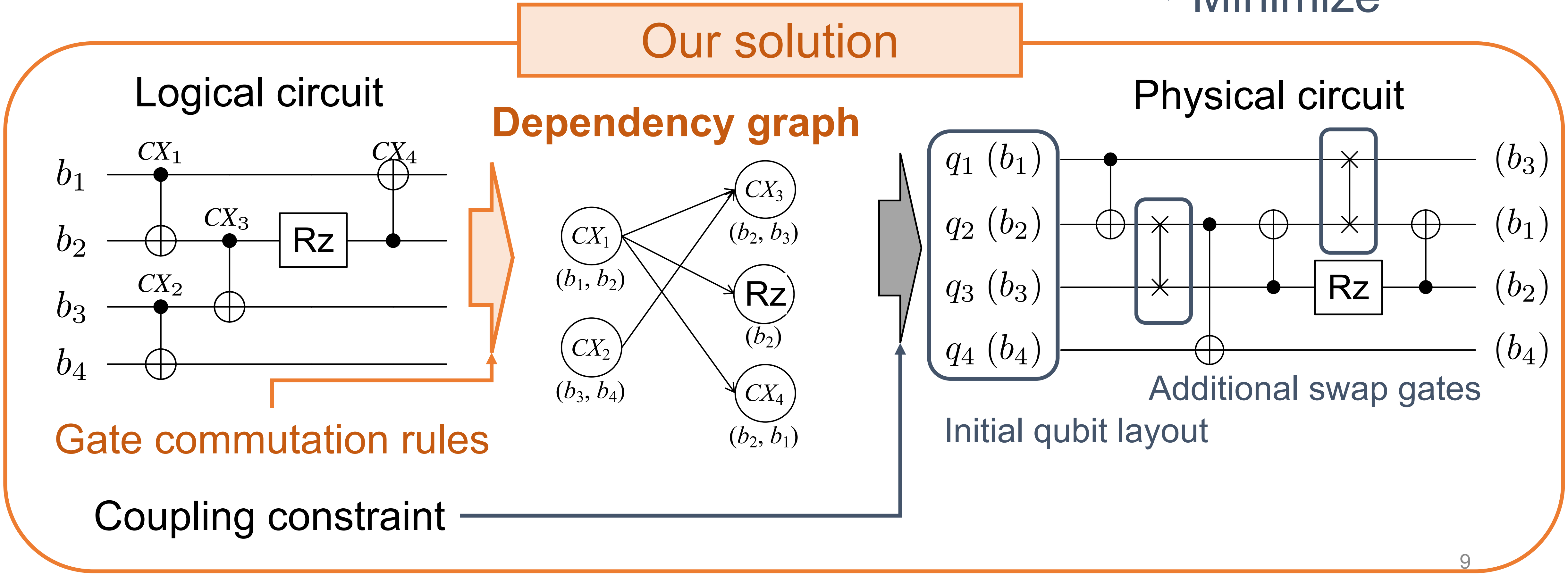


Logical qubits are swapped

CNOT(b_3, b_4) is now executable as CNOT(q_2, q_4)

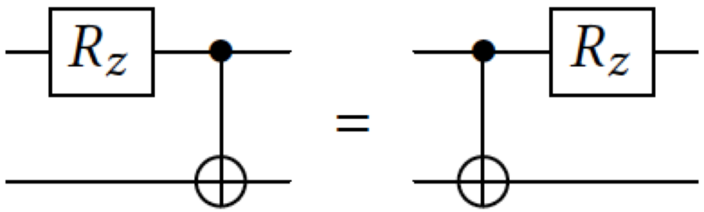
Minimum Swap Gate Mapping (MSGM)

- Logical circuit + **Gate commutation rules** + Coupling constraint -> Physical circuit (Initial qubit layout + Additional swap gates)
  ↳ Minimize

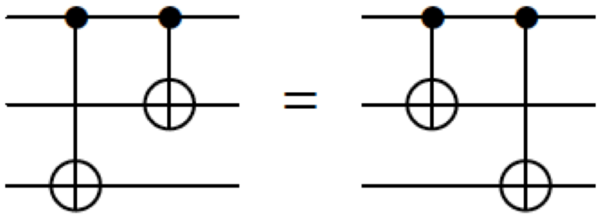


Gate commutation rules

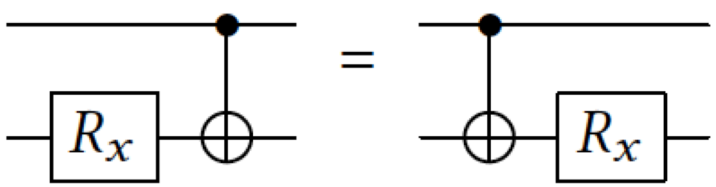
We take into account 4 commutation rules:



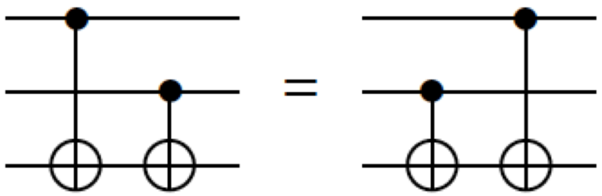
(a) R_z−control



(b) Control–control



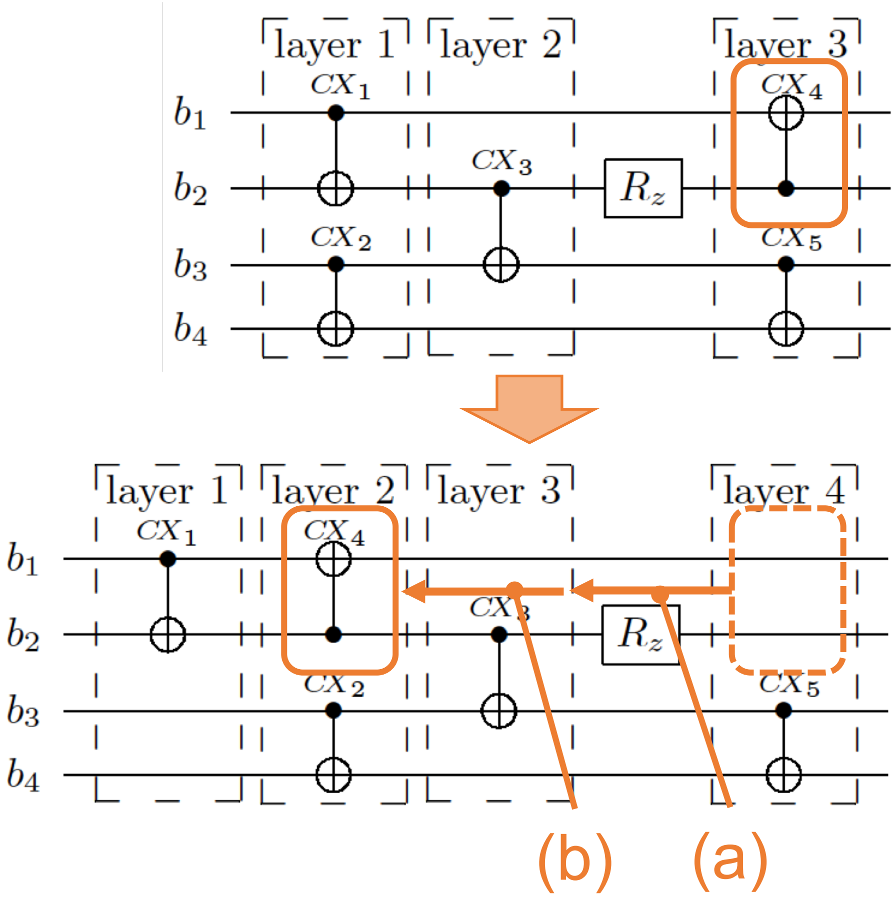
(c) R_x−target



(d) Target–target

Two consecutive gates are commutative ⇔
They can be exchanged without changing what they compute.

Ex) Equivalent conversion
considering commutation rules



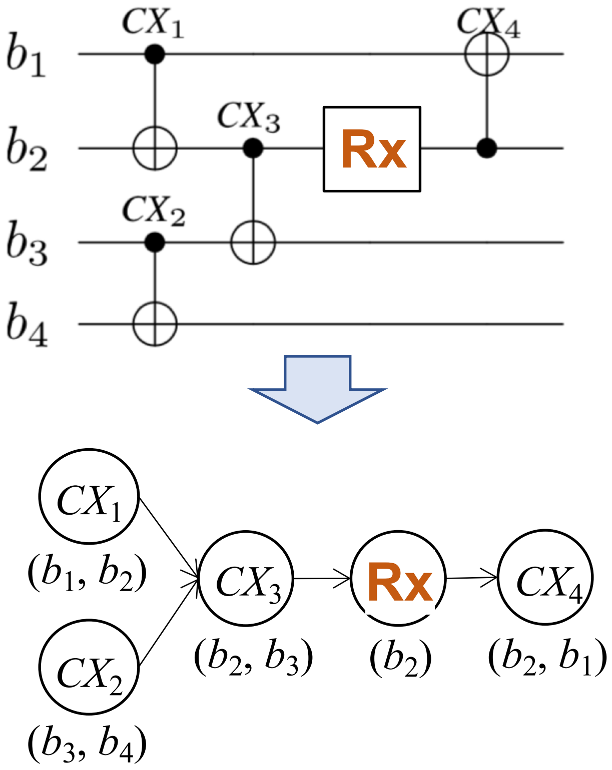
Dependency graph

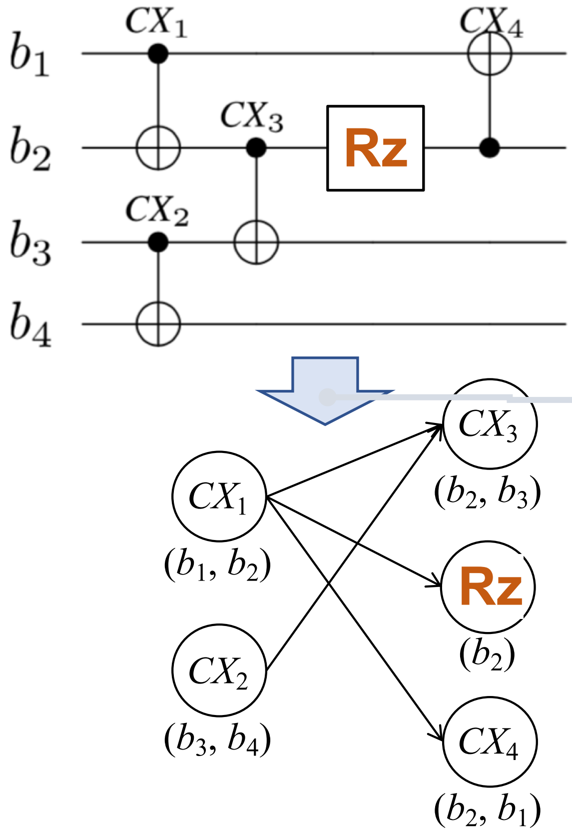
Dependency graph represents the precedence relation of gates in a circuit.

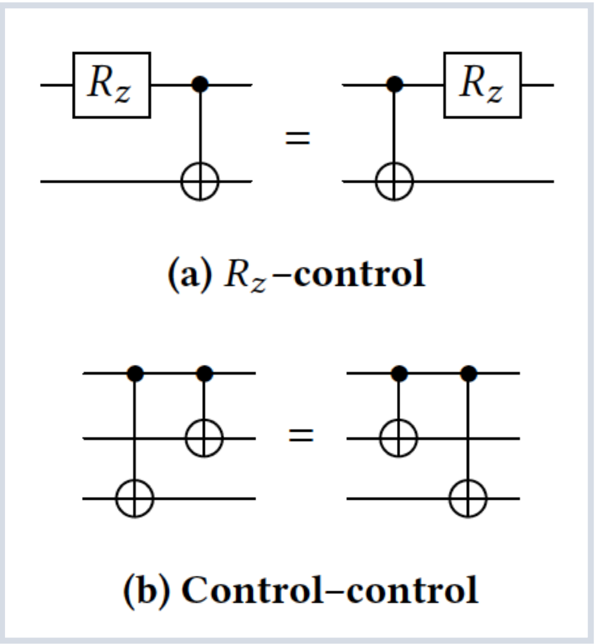
Node ⇔ Gate

Edge ⇔ Dependency

Gate u precedes v ⇔ Path from u to v exists



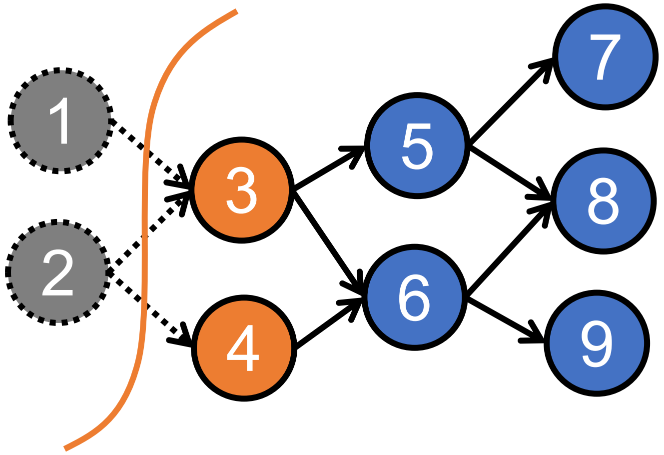




Blocking Gates

Our algorithms maintain their progress by **blocking gates**.

Blocking gates are **leading unresolved gates** in dependency graph (for a current qubit layout).



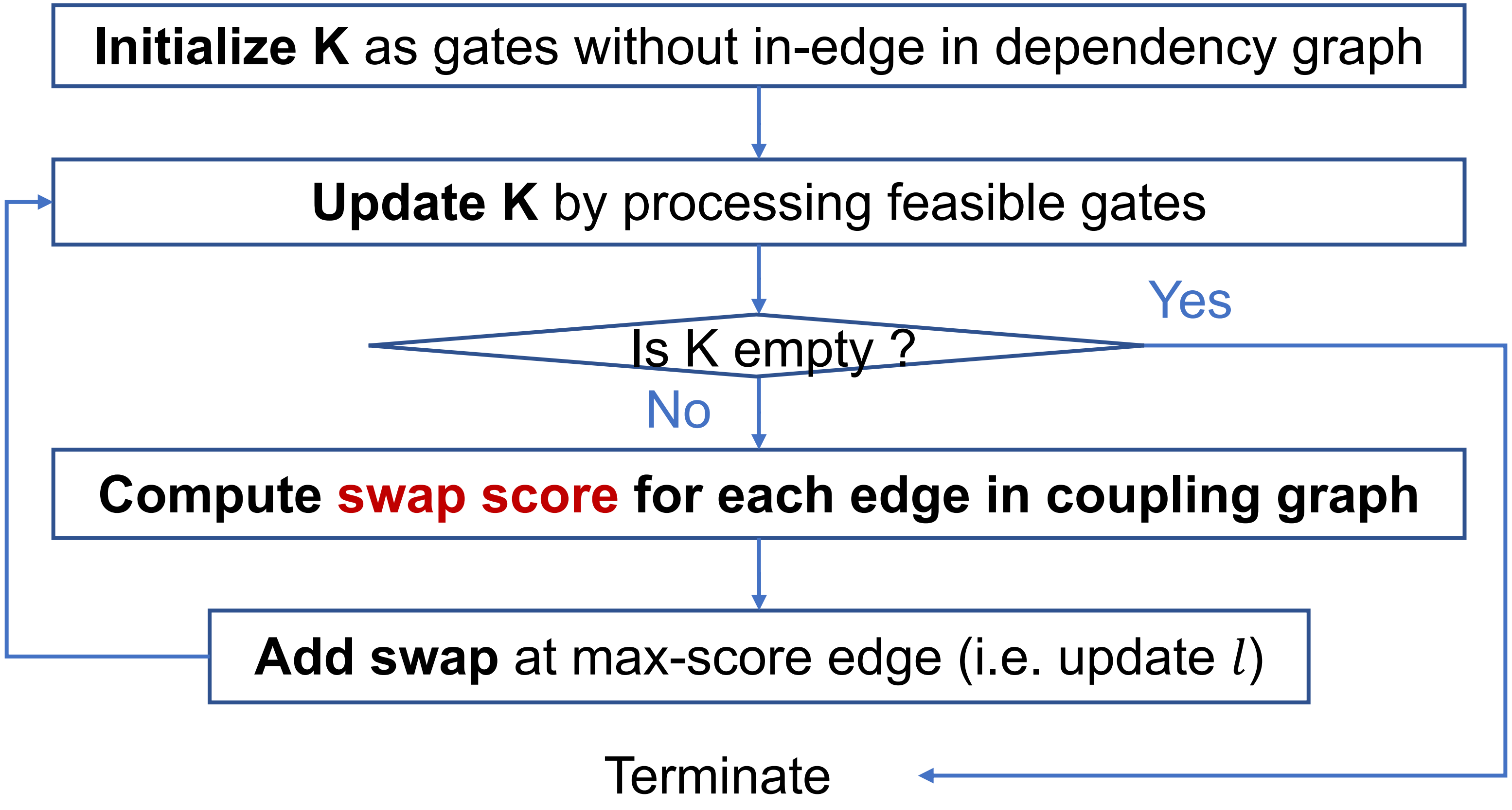
Resolved gates = {1, 2}

Blocking gates = {3, 4}

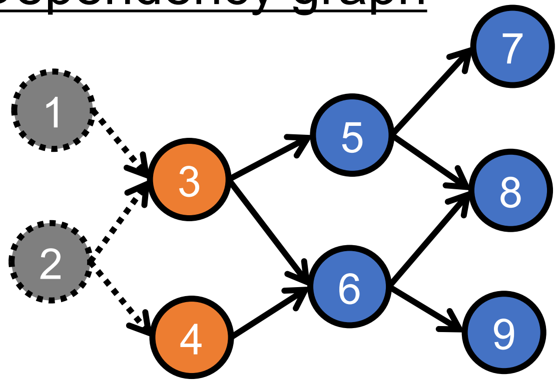
Unresolved gates = {3, 4, 5, 6, 7, 8, 9}

Heuristic algorithm (Outline)

- Maintains layout l and blocking gates K
- Assumes an initial layout is given
- Selects a qubit pair to be swapped based on its **swap score**



Dependency graph



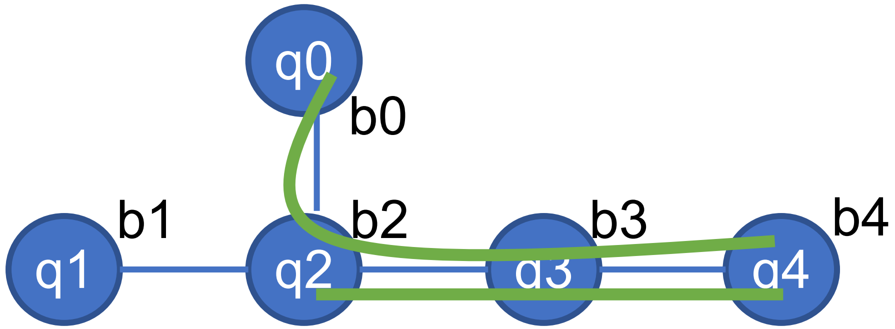
(Current) blocking gates K

Heuristic algorithm (Details)

For each edge i.e. coupled physical qubits (i, j),
swap score of (i, j) := current **cost** – **cost** after swap (i, j)

cost := sum of (**weighted**) **shortest path lengths** on coupling graph between acting qubits for all unresolved gates

Coupling graph



Layout l = (b0, b1, b2, b3, b4) → (q0, q1, q2, q3, q4)
Blocking gates K = {CNOT(b0, b4)}
Unresolved gates = {CNOT(b0, b4), CNOT(b2, b4)}

CNOT(b0,b4) CNOT(b2,b4)

current cost = (**1.0**) x 3 + (**0.5**) x 2

cost after swap (i, j) =
- (q0, q2) => (1.0) x 2 + (0.5) x 3
- (q1, q2) => (1.0) x 3 + (0.5) x 3
- (q2, q3) => (1.0) x 3 + (0.5) x 1
- (q3, q4) => (1.0) x 2 + (0.5) x 1

Computational experiment: Setting

We compared the numbers of additional swap gates of **our heuristic** with those of two state-of-the-art algorithms.

QRAND: A randomized heuristic algorithm implemented in QISKit 0.5.4
ZPW: A$*$-based heuristic search algorithm proposed by Zulehner, Paler, Wille (2018)

We set the initial qubit layout for our heuristic and QRAND to b_i -> q_i for all i.

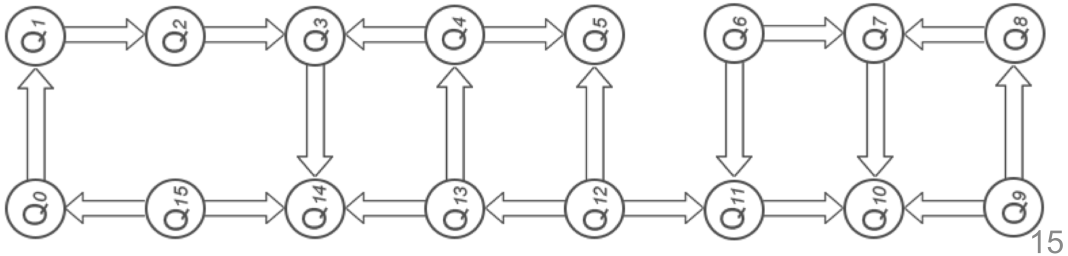
<u>Dataset</u>

Circuits originated from RevLib benchmark

We chose 44 circuits with #qubits ≥ 10 and #gates ≤ 50, 000 from the circuits available at http://iic.jku.at/eda/research/ibm_qx_mapping

Coupling graphs

IBM Q 16 Rueschlikon V1.0.0 (ibmqx3)



Evaluation of heuristic algorithm

Our algorithm outperformed QRAND and ZPW for all instances.

- #swaps decreased by 45.5% from QRAND, 23.8% from ZPW on average

Numbers of additional swap gates (for circuits with 10 qubits)

Circuit name	#qubits	#gates	QRAND	ZPW	Proposed
mini_alu_305	10	173	80	46	**40**
qft_10	10	200	82	40	**33**
sys6-v0_111	10	215	116	67	**46**
rd73_140	10	230	100	58	**49**
ising_model_10	10	480	18	14	**12**
rd73_252	10	5,321	2,054	1,541	**1,212**
sqn_258	10	10,223	4,060	2,867	**2,254**
sym9_148	10	21,504	8,001	5,907	**4,456**
max46_240	10	27,126	10,833	8,012	**5,905**

Summary

- **Considering gate commutation rules in the formulation** of quantum circuit mapping is significant.
- Dependency graph helps us develop better algorithms:
  Our heuristic algorithm performs very well in the experiment.

Future work

- Finding better initial qubit layouts
- Considering other cost functions
  - Depth
  - Fidelity