Quantum Circuit Compilers Using Gate Commutation Rules

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


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 a 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**

IN: Logical circuit

OUT: Physical circuit

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 is a sequence of elementary quantum gates.

We consider a universal gate set \( \{ Rx, Rz, \text{CNOT} \} \).

- One-qubit rotation gates: \( Rx \), \( Rz \)

- Controlled-NOT gate (\( \text{CNOT} \))

Ex) Quantum circuit
Coupling constraint

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

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

Example (Initial qubit layout: $b_i \rightarrow q_i$ for $i = 1,2,3,4$)

- Not executable
- Add $\text{SWAP}(q_2, q_3)$
- Logical qubits are swapped
- $\text{CNOT}(b_3, b_4)$ is now executable as $\text{CNOT}(q_2, q_4)$
Minimum Swap Gate Mapping (MSGM)

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

Our solution

Logical circuit

![Logical circuit diagram](image)

**Gate commutation rules**

**Coupling constraint**

Dependency graph

![Dependency graph](image)

Physical circuit

![Physical circuit diagram](image)

**Initial qubit layout**

**Additional swap gates**

Minimize
Gate commutation rules

We take into account 4 commutation rules:

1. **(a) $R_z$–control**
   
   
   
   

2. **(b) Control–control**
   
   
   

3. **(c) $R_x$–target**
   
   

4. **(d) Target–target**
   
   

Ex) Equivalent conversion considering commutation rules:

Two consecutive gates are commutative $\Leftrightarrow$

They can be exchanged without changing what they compute.
Dependency graph

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

Node $\leftrightarrow$ Gate
Edge $\leftrightarrow$ Dependency

Gate $u$ precedes $v$ $\iff$ 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

Initialize \( K \) as gates without in-edge in dependency graph

Update \( K \) by processing feasible gates

Is \( K \) empty?

Yes

No

Compute swap score for each edge in coupling graph

Add swap at max-score edge (i.e. update \( l \))

Terminate
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*

```
q0 -- b0 |       |
|       |       |       |       |
q2 -- b1 | q1 -- b2 | q3 -- b3 | q4 -- b4
```

Current cost = \((1.0) \times 3 + (0.5) \times 2\)

Cost after swap \((i, j)\) =

- \((q0, q2)\) => \((1.0) \times 2 + (0.5) \times 3\)
- \((q1, q2)\) => \((1.0) \times 3 + (0.5) \times 3\)
- \((q2, q3)\) => \((1.0) \times 3 + (0.5) \times 1\)
- \((q3, q4)\) => \((1.0) \times 2 + (0.5) \times 1\)

Layout \(l = (b0, b1, b2, b3, b4) \rightarrow (q0, q1, q2, q3, q4)\)

Blocking gates \(K = \{\text{CNOT}(b0, b4)\}\)

Unresolved gates = \{\text{CNOT}(b0, b4), \text{CNOT}(b2, b4)\}
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 $b_i \rightarrow q_i$ for all $i$.

Dataset

Circuits originated from RevLib benchmark

We chose 44 circuits with $\#\text{qubits} \geq 10$ and $\#\text{gates} \leq 50,000$ 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)

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