A Binary Integer Programming-Based Method for Qubit Mapping in Sparse Architectures

Hui Jiang[1], Jianling Fu[2], Yuxin Deng[1*], Jun Wu[1]

[1*]Shanghai Key Lab of Trustworthy Computing, East China Normal University, Shanghai, 200063, China.
[2]Shanghai Institute for AI Education, East China Normal University, Shanghai, 200063,China.

*Corresponding author(s). E-mail(s): yxdeng@sei.ecnu.edu.cn;
Contributing authors: yhq_jh@126.com; scssfjl2015@126.com;
51215902063@stu.ecnu.edu.cn;

Abstract

It is a current trend of sparse architectures employed for superconducting quantum chips, which have the advantage of low coupling and crosstalk properties. Existing qubit mapping algorithms do not take the sparsity of quantum architectures into account. To this end, we propose a qubit mapping method based on binary integer programming, called QMBIP. First, we slice a given quantum circuit by taking into account the sparsity of target architectures. Then, the constraints and the objective function are formulated and rendered to the binary integer programming problem by matrix transformation. The behavior of a SWAP gate is characterized by an elementary row transformation on the mapping matrix between the physical and logical qubits. To reduce the search space, we introduce path variables and isomorphic pruning, as well as a look-ahead mechanism. Finally, we compare with typical qubit mapping algorithms such as SABRE and SATMAP on the sparse architectures ibmq_sydney, ibmq_manhattan, ibmq_singapore, and a dense architecture ibmq_tokyo. Experiments show that QMBIP efectively maintains the fdelity of the compiled quantum circuits. For example, on ibmq_sydney, the fdelity of the quantum circuits compiled by our approach outperforms SABRE and SATMAP by 53.9% and 46.8%, respectively.

Keywords: qubit mapping, sparse architecture, binary integer programming

1 Introduction

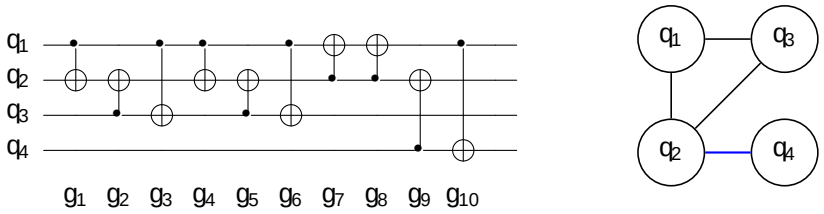
Over the past four decades, quantum computing has been evolving at a rapid pace. In 1980, Paul Beniof proposed the quantum mechanical model of Turing machines [1]. During the last two decades, tech giants have developed prototypes of quantum computers aiming to solve specifc tasks. For example, in late 2023, IBM released the frst-ever 1,000-qubit superconducting quantum chip \Condor" [1].

There are several ways of building a quantum computer, e.g., ion-trap quantum computers [2], superconducting quantum circuits [3] and photonic quantum devices [4]. Superconducting quantum circuits are widely investigated because they enjoy good scalability, but in which a qubit is only connected to its adjacent qubits. Table 1 shows the evolution of the topology and the average qubit connectivity for

2 Preliminaries

In this section, we introduce some notions and notations of quantum computing. Let $\mathbb{N}$ and $\mathbb{C}$ denote the set of all natural and complex numbers, respectively.

Classical information is stored in bits, while quantum information is stored in qubits. Besides two basic states $|0\rangle$ and $|1\rangle$, a qubit can be in any linear superposition state $|\phi\rangle = a|0\rangle + b|1\rangle$, where $a, b \in \mathbb{C}$ satisfy the condition $|a|^2 + |b|^2 = 1$. The state $|\phi\rangle$ is in the state $|0\rangle$ with probability $|a|^2$ and in the state $|1\rangle$ with probability $|b|^2$. In a quantum circuit, each line represents a wire. The wire does not necessarily correspond to a physical wire but may represent the passage of time or a physical particle that moves from one location to another through space. The scale of a quantum circuit is the number of 2-qubit gates. The interested reader can fnd more details of these gates from the standard textbook [23].



(a) (b)

Fig. 1 (a) A quantum circuit; (b) The frst interactions between any two qubits

The architecture of a quantum chip called a coupling graph is denoted by $G = (Q, E)$, where Q is a set of physical qubits and E is an edge set between physical qubits. The letter q (resp. Q) denotes a logical (resp. physical) qubit set, where $q = \{q_i\}_{i \in I}$ and $Q = \{Q_j\}_{j \in J}$ for some index sets I and J. A path $Q_1 \to Q_2 \to \cdots \to Q_n$ consists of a set of transformations $Q_i \to Q_{i+1}$, $(1 \le i \le n-1)$ with each transformation denoting a SWAP gate. A CX gate is a pair $g = (q_c, q_t)$ for $c \neq t$ and $c, t \in I$, where q_c is the control qubit and q_t is the target qubit. The mapping between the physical qubits and the logical qubits is a $|Q| \times |q|$-dimensional matrix M, where each row (resp. column) corresponds to a physical (resp. logical) qubits. The entry $M_{j,i} = 1$ means that the logical qubit q_i is mapped to the physical qubit Q_j, otherwise $M_{j,i} = 0$, satisfying $\|M_i\|_1 = 1$. The symbol $\|\cdot\|_1$ denotes the 1-norm of a vector. The set $R(q_i) = \{Q_j \mid M_{j,i} \neq 0\}$ consists of the candidate physical qubits mapped by logical qubit q_i. The symbol $R(g)$ denotes a set $\{(Q_1, Q_2) \mid Q_1 \in R(q_c), Q_2 \in R(q_t)$, and $Q_1 \neq Q_2\}$, which contains all the pairs of candidate physical qubits for the CX gate g. The frst interaction represents the frst CX gate acting on two qubits. In Fig. 1 (b), the black edges $(q_1, q_2), (q_1, q_3), (q_2, q_3)$ correspond to the CX gate g_1, g_3, g_2 in Fig. 1 (a), respectively, which are in the first slice. The blue edge (q_2, q_4) corresponds to the gate g_9. Fig. 2 shows the architecture of the 27-qubit quantum chip `ibmq_sydney`, where the nodes represent the physical qubits and the edges represent the interactions between

two physical qubits. The label on each edge is the fdelity of aCX gate operating on the endpoints. The fdelity of a quantum circuit is the product of the fdelities of all the gates.

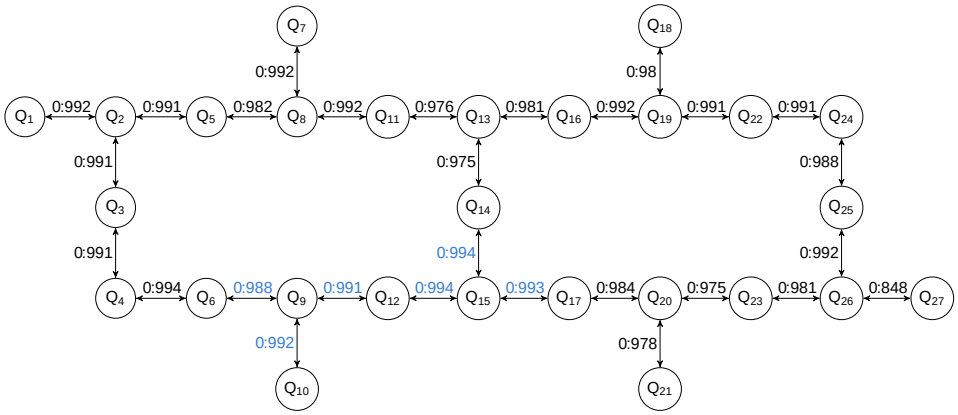


Fig. 2 The architecture of the 27-qubit quantum chip ibmq_sydney

The general form of a binary integer programming problem is as follows,

$$\text{minimize} \quad \sum_{i=1}^{n} c_i x_i \tag{1}$$

$$\text{such that:} \quad \sum_{i=1}^{n} a_i x_i = b \text{ and } \bigwedge_{i=1}^{n} (x_i \in \{0,1\}) \tag{2}$$

where the constant coefficients a_i, c_i, b belong to $[0,1]$, (1) is the objective function and (2) is the constraint. To encode the constraints of a slice into a dependency chain of variables, we define a boolean function,

$$f(x_1, x_2, \ldots, x_k) = x_1 \wedge x_2 \wedge \cdots \wedge x_k$$

where $x_1, x_2, \ldots, x_k \in \{0,1\}$ are binary variables. Furthermore, when the input consists of matrices and constants, the general form of f is as follows,

$$f(X^1, \ldots, X^l, c, x_1, \ldots, x_k) = \bigwedge_{ij} f(X^1_{ij}, \ldots, X^l_{ij}, c, x_1, \ldots, x_k)$$

where $c = c_1, c_2, \ldots, c_n \in \{0,1\}$ are constants and $X^1, \ldots, X^l \in \{0,1\}^{|Q_j| \times |Q_j|}$. Note that the boolean constraints can be linearized.

3 Qubit Mapping

The qubit mapping problem mainly consists of initial mapping and qubit mapping adjustment, which is defined as follows.

Defnition 3.1 (The Qubit Mapping Problem [5]). Input: a coupling graph $G = (Q, E)$ and a list $\Psi = (q \times q)^n; n \geq 1$, of n control relations between logical qubits q, an integer $K_C \geq 0$, a list of allowed quantum transformations θ, and a function $C : \theta \to \mathbb{N}$ that gives the cost to implement each transformation. Output: Yes, if we can produce a version of Ψ that complies with G with transformations whose total cost does not exceed K_C.

3.1 Slicing

The triangle pattern is our criterion for slicing in the topology of a quantum circuit. The reasons are as follows.

- The time cost of solving a BIP problem encoded by the whole circuit is mainly afected by the scale of quantum circuits and target architectures.
- Compiling a quantum circuit on a sparse architecture typically needs to insert some SWAP gates as soon as the topology of the CX gates forms a triangle.
- There are many consecutive slices with isomorphic interactions on the same qubit set, called similar slices, which are the cornerstone of the look-ahead mechanism.

Example 3.1. The topology of the quantum circuit in Fig. 1 (a) is shown in Fig. 3. The green nodes represent incoming wire nodes and the red ones represent outgoing wire nodes. The remaining nodes are 2-qubit gates. The label on a directed edge is the operation qubit of the successor node. The in-degree of each gate is equal to the out-degree. We slice the circuit topology based on the triangles marked by the blue lines in Fig. 3. The set of slices is $\{\{g_1, g_2, g_3\}, \{g_4, g_5, g_6\}, \{g_7, g_8, g_9, g_{10}\}\}$. Note that the gates g_7 and g_8 have the same operational qubits, which are sliced into the third slice.

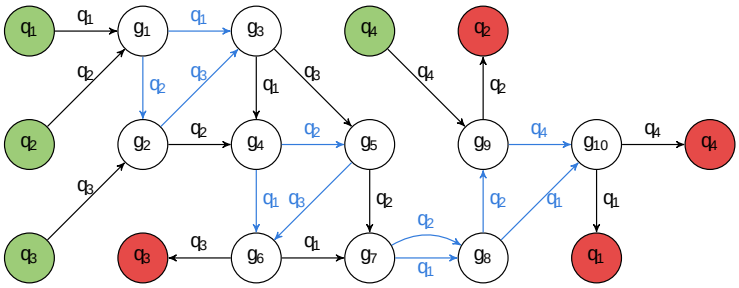


Fig. 3 The topology of the quantum circuit in Fig. 1 (a)

3.2 Initial Mapping

For diferent physical architectures, there are diferent initial mapping methods. For example, weighted dependence graph [5] and subgraph isomorphism of the whole quantum circuit [8] are related to the degree of nodes, which are suitable for dense architectures such as ibmq_tokyo with an average qubit connectivity of 3.9. In sparse architectures with an average qubit connectivity of 2.1, the initial mapping may be

frequently changed during the process of circuit compilation, even though the interactions of all gates are considered in the initial mapping. Thus, we only consider the frst interactions between any two qubits for constructing an initial mapping. First, we fnd the set of maximal isomorphic subgraphs between the interactions and the target architecture. Then, the matched isomorphic subgraphs are fltered by the sum of the fdelities of each matched physical qubit and its neighbors. Here, we take the sum of those fdelities, which intuitively mimics the degrees of the relevant nodes in the coupling graph. Finally, the remaining nodes are mapped to the physical nodes with the closest degree on the coupling graph.

Algorithm 1 Constructing the initial mapping

Input: The sliced quantum circuit C, the edge set E of the target architecture.
Output: A qubit mapping matrix M.

```
G_c  compute the frst interactions between any two qubits in quantum circuit C;
G_e  compute a maximal isomorphic mapping between G_c and E, fltered by the
     maximum sum of the fdelities of each matched physical qubit and its neighbors;
for all the unmapped logical qubit q do
    N_q  compute the mapped physical qubits of the neighbors of q in the
         interactions G_c;
    N_p  compute the unmapped neighbors of the physical qubits in N_q;
    G_e  G_e [ {q $ n | n 2 N_p and with the closest degree to q};
end for
M  convert the mapping G_e to a 0{1 matrix;
return M;
```

Example 3.2. Fig. 1 (b) shows the interactions of the frst slice $\{(q_2, q_1), (q_3, q_2), (q_1, q_3)\}$ interacting with the remaining node q_4, where the blue line is the frst interaction (q_4, q_2) between the remaining qubit q_4 and the frst slice. There are some maximum common subgraphs $\{\{q_1 \$ Q_1, q_2 \$ Q_2, q_4 \$ Q_3\}, \{q_1 \$ Q_2, q_2 \$ Q_3, q_4 \$ Q_4\}, \cdots\}$ between the interactions and the target architecture ibmq_sydney. We use the largest sum of degrees to flter out the maximum common subgraphs with the highest fdelity. The sum of fdelities of the partial mapping $\{q_1 \$ Q_9, q_2 \$ Q_{12}, q_4 \$ Q_{15}\}$ is $2 \times (0.988 + 0.992 + 0.991 + 0.991 + 0.994 + 0.994 + 0.994 + 0.993) = 15.874$, which is the sum of each label of mapped nodes $\{Q_9, Q_{12}, Q_{15}\}$ in Fig. 2. The qubit q_3 is still not mapped, which interacts with the mapped qubits q_1 and q_2. Thus the qubit q_3 should be mapped to one of the unmapped neighbors $\{Q_6, Q_{10}\}$ of physical qubits $R(q_1) \cup R(q_2) = \{Q_9, Q_{12}\}$. The qubit q_3 is mapped to the physical qubit Q_6 with the closest degree. The initial mapping becomes $\{q_1 \$ Q_9, q_2 \$ Q_{12}, q_3 \$ Q_6, q_4 \$ Q_{15}\}$.

3.3 Constriants

The constraints of qubit mapping refer to the injective mapping from logical qubits to physical ones and the logical gates are executable in physical devices. The executable constraints focus on the interactions between logical gates and the interactions between physical qubits.

3.3.1 Injective Constraint

The mapping from logical qubits to physical ones is injective in which the sum of each column of the mapping matrix M is 1, and the sum of each row is not greater than 1, i. e.,

$$\sum_{i=1}^{|Q|} M_{j,i} = 1 \text{ and } \sum_{j=1}^{|Q|} M_{j,i} \leq 1.$$

Suppose a SWAP gate acts on a pair of qubits (Q_m, Q_n). Its efect is to exchange the m-th and n-th rows of matrix M. Based on this observation, we use a $|Q| \times |Q|$-dimensional elementary matrix to describe a SWAP gate.

Example 3.3. We continue to consider the Example 3.2, focusing on the efect of a SWAP gate on the initial mapping $\{q_1 \$ Q_9, q_2 \$ Q_{12}, q_3 \$ Q_6, q_4 \$ Q_{15}\}$. A SWAP gate acts on the qubit Q_9 and Q_{12}, which is refected by the initial mapping matrix $M_{27\times 4}$ multiplied by an elementary matrix $P_{27\times 27}$,

$$P = \begin{array}{c} \\ 0 \\ \\ \\ \\ \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} & 9 & 10 & 11 & 12 \end{array} \\ \begin{pmatrix} \cdots & \vdots & \vdots & \vdots & \vdots & \cdots \\ \cdots & 0 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 1 & \cdots \\ \cdots & 0 & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & 0 & \cdots \\ \cdots & 1 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \end{array} \begin{array}{c} \\ 1 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}$$

$$; M = \begin{array}{c} \\ 0 \\ \\ \\ \\ \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \end{array} \begin{array}{c} \\ 1 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}$$

$$; PM = \begin{array}{c} \\ 0 \\ \\ \\ \\ \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \end{array} \begin{array}{c} \\ 1 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} ;$$

3.3.2 Executable Constraint

Note that 2-qubit gates can only be executed on two adjacent physical qubits and a sequence of SWAP gates can exchange the state of a qubit to any connected physical qubit. We use the gray code algorithm [24] to compute the set P of shortest paths that can exchange the states of a qubit pair $(Q_m, Q_n) \in R(g)$ to any target one $(Q_s, Q_r) \in E$ fltered by the fdelities, where $g = (q_c, q_t)$ and $1 \leq m, n, s, r \leq |Q|$. Each path $P_k \in P$ can be represented by a matrix P_k, which is the product of a sequence of elementary matrices $P_k = P_{k,|P_k|} \cdots P_{k,2} P_{k,1}$, that acts on the mapping matrix M and results in a mapping matrix M_k^0 after the adjustments,

$$M_k^0 = P_k M. \tag{3}$$

Each shortest path is considered as an option. We use a binary variable $x \in \{0,1\}$ to indicate that either the path is chosen ($x = 1$) or not ($x = 0$). The path $P_k \in P$ moves the state of qubits Q_m, Q_n to the qubits Q_s, Q_r, which is indicated by the variable $x_k \in \{0,1\}$, satisfying $\|x\|_1 = 1$, for $x = [x_1, \ldots, x_{|P|}]^\top \in \{0,1\}^{|P|}$. Whether the path $P_k \in P$ is chosen depends on not only the variable $x_k = 1$, but also the truth value of the current mapped qubit pair in the mapping matrix, i.e. $M_{m,c} = 1$ and $M_{n,t} = 1$. In summary, the path variable v_k is defined as

$$v_k = f(M_{m,c}, M_{n,t}, x_k). \tag{4}$$

Example 3.4. For the quantum circuit in Fig. 1 (a), the gate g_1 is executable, while the second gate g_2 is not executable. The current qubit pair set of g_2 is $R(g_2) = \{(Q_6, Q_{12})\}$, and the target one is in the set E. We get the set of shortest paths $P = \{P_1, P_2\}$ filtered by the fidelity limitation 0.956, where $P_1 = \{Q_{12} \to Q_9 \to Q_6\}$, $P_2 = \{Q_6 \to Q_9 \to Q_{12}\}$. The binary variables x_1 and x_2 represent the paths P_1 and P_2, respectively, satisfying the condition $x_1 + x_2 = 1$. The path variable of P_1 is $v_1 = f(M_{6,3}, M_{12,2}, x_1) = f(1, 1, x_1)$, where $M_{6,3} = M_{12,2} = 1$. Similarly, the path variable of P_2 is $v_2 = f(M_{6,3}, M_{12,2}, x_2) = f(1, 1, x_2)$. If the constraints $x_1 = 0$ and $x_2 = 1$ hold then we also have $v_1 = 0$ and $v_2 = 1$, which means that the state of qubit pair (Q_6, Q_{12}) is swapped to the qubit pair (Q_9, Q_{12}) following the path P_2.

For each path $P_1, \ldots, P_{|P|}$, there is a sequence of matrix $M'_1, \ldots, M'_{|P|}$ corresponding to it, respectively. The matrix M''_k is combined by the path variable v_k and the matrix M'_k via the function f,

$$M''_k = f(M'_k, v_k) = \begin{cases} M'_k & \text{if } v_k = 1; \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

When the value of the path variable v_k is 1, the matrix M''_k is equivalent to M'_k, otherwise M''_k is set to be 0. The matrices $M''_1, \ldots, M''_{|P|}$ are gathered as the mapping matrix $M'' = \sum_{k=1}^{|P|} M''_k$. Note that all of the path variables satisfy $\|v\|_1 = 1$, where $v = [v_1, \ldots, v_{|P|}]^\top \in \{0,1\}^{|P|}$.

Example 3.5. Now let us continue to consider the Example 3.4. We only encode the executable constraint of the gates. For the gate g_2, the paths P_1 and P_2 can be represented by the elementary row transformations P_1 and P_2, respectively,

$$P_1 = \begin{array}{c} \\ 9 \\ 10 \\ 11 \\ 12 \\ \\ 27 \times 27 \end{array} \begin{array}{c} \begin{array}{ccccc} 9 & 10 & 11 & 12 & \end{array} \\ \begin{array}{ccccc} 0 & \cdots & \cdots & \cdots & 1 \end{array} \\ \begin{array}{ccccc} \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & \cdots \\ \cdots & 1 & 0 & 0 & \cdots \end{array} \\ \begin{array}{ccccc} \ddots & \vdots & \vdots & \vdots & \ddots \end{array} \end{array} \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \\ \\ \\ \\ 1 \\ \\ \end{array}$$

$$; P_2 = \begin{array}{c} \\ 6 \\ 7 \\ 8 \\ 9 \\ \\ 27 \times 27 \end{array} \begin{array}{c} \begin{array}{ccccc} 6 & 7 & 8 & 9 & \end{array} \\ \begin{array}{ccccc} 0 & \cdots & \cdots & \cdots & 1 \end{array} \\ \begin{array}{ccccc} \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & \cdots \\ \cdots & 1 & 0 & 0 & \cdots \end{array} \\ \begin{array}{ccccc} \ddots & \vdots & \vdots & \vdots & \ddots \end{array} \end{array} \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \\ \\ \\ \\ 1 \\ \\ \end{array} ;$$

After the elementary row transformation P_1 and combining with corresponding path variable v_1, we get the mapping matrices $M_1^0 = P_1 M$ and $M_1^{00} = f(M_1^0, v_1)$,

$$M_1^0 = \begin{array}{c} \\ 0 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ @ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \\ 27 \times 4 \end{array} \begin{array}{c} \\ 1 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}$$

$$; M_1^{00} = \begin{array}{c} \\ 0 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ @ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & f(1; v_1) & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & f(1; v_1) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ f(1; v_1) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & f(1; v_1) \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \\ 27 \times 4 \end{array} \begin{array}{c} \\ 1 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} :$$

Similarly, after the elementary row transformation P_2 and combining with corresponding path variable v_2, we get the matrices M_2^0 and $M_2^{00} = f(M_2^0, v_2)$, respectively,

$$M_2^0 = \begin{array}{c} \\ 0 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ @ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \\ 27 \times 4 \end{array} \begin{array}{c} \\ 1 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}$$

$$; M_2^{00} = \begin{array}{c} \\ 0 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ @ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ f(1; v_2) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & f(1; v_2) & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & f(1; v_2) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & f(1; v_2) \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \\ 27 \times 4 \end{array} \begin{array}{c} \\ 1 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} :$$

Finally, we get the mapping matrix M^{00} by combining with the elementary row transformations P_1 and P_2,

$$M^{00} = M_1^{00} + M_2^{00} = \begin{array}{c} \\ 0 \\ \vdots \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ f(1;v_2) & 0 & f(1;v_1) & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & f(1;v_1) & f(1;v_2) & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ f(1;v_1) & f(1;v_2) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & f(1;v_1) + f(1;v_2) \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}_{27 \times 4} \end{array} \begin{array}{c} \\ 1 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} :$$

where the entry $f(1;v_1) + f(1;v_2)$ of matrix M^{00} equals to 1, since either $v_1 = 1$ or $v_2 = 1$ holds. So are the two nonzero entries $f(1;v_1)$ and $f(1;v_2)$ in the columns of matrix M^{00}. Thus, the constraints for the quantum gate g_2 are as follows,

$$\begin{cases} x_1 + x_2 = 1; \\ v_1 + v_2 = 1; \\ x_1; x_2; v_1; v_2 \in \{0;1\}. \end{cases}$$

For the third gate $g_3 = (q_1; q_3)$, the qubits q_1 and q_3 are mapped to $R(q_1) = \{Q_6; Q_{12}\}$ and $R(q_3) = \{Q_6; Q_9\}$, respectively. The set of currently mapped qubit pairs is $R(g_3) = \{(Q_6; Q_9); (Q_{12}; Q_6); (Q_{12}; Q_9)\}$. Then there are four paths $P = \{P_3 = \{Q_6 \to Q_9\}; P_4 = \{Q_{12} \to Q_9 \to Q_6\}; P_5 = \{Q_6 \to Q_9 \to Q_{12}\}; P_6 = \{Q_{12} \to Q_9\}\}$ filtered by the fidelity limitation 0.956. The path variables are as follows,

$$\begin{aligned} v_3 &= f(M_{6;1}; M_{9;3}; x_3) = f(f(1;v_2); f(1;v_2); x_3); \\ v_4 &= f(M_{12;1}; M_{6;3}; x_4) = f(f(1;v_1); f(1;v_1); x_4); \\ v_5 &= f(M_{12;1}; M_{6;3}; x_5) = f(f(1;v_1); f(1;v_1); x_5); \\ v_6 &= f(M_{12;1}; M_{9;3}; x_6) = f(f(1;v_1); f(1;v_2); x_6) = 0: \end{aligned} \tag{6}$$

Finally, the constraints of the gate g_3 are as follows,

$$\begin{cases} x_3 + x_4 + x_5 + x_6 = 1; \\ v_3 + v_4 + v_5 + v_6 = 1; \\ x_3; x_4; x_5; x_6; v_3; v_4; v_5; v_6 \in \{0;1\}. \end{cases}$$

3.4 Objective Function

The most accurate metric for evaluating a compiled circuit is the fdelity of the generated circuit, which is afected primarily by the error rate of the gates in the circuit and the error caused by decoherence. The objective function considers only the fdelity of each path,

$$\text{maximize} \sum_{i=1}^{n} \sum_{k=1}^{|P_i|} w_k v_k ;$$

where n is the number of gates in a slice and $|P_i|$ is the size of the shortest path set for the i-th gate. The weight w_k is the fdelity of the path indicated by the variable v_k. By simply modifying the label of each edge on the architecture to a constant e, where $0 < e < 1$, the objective function is extended to the least number of 2-qubit gates inserted. We summarize the encoding procedure in Algorithm 2.

Algorithm 2 Encoding into binary integer programming

Input: A slice of the quantum circuit G, a qubit mapping matrix M, the edge set E of the target architecture and the fdelity limitation W.

Output: A constraint set T and the objective function F.

1: $F \quad 0$ and $T \quad ; \quad ;$
2: for all g in G do
3: if g is a 2-qubit gate then
4: $M^{00} \quad 0;$
5: $P \quad$ compute the set of the shortest paths from each qubit pair $(Q_m, Q_n) \in R(g)$ to any edge in E fltered by the fdelity limitation W;
6: let $x = [x_1; \ldots; x_{|P|}]^>$ be a vector of binary variables and $v = [v_1; \ldots; v_{|P|}]^>$ be a vector of path variables for each path $P_k \in P$ as defned in (4);
7: for $k \quad 1$ to $|P|$ do
8: $M_k^0 \quad M$;
9: $Q_m, Q_n \quad$ the currently mapped qubit pairs of P_k;
10: let w be the fdelity of the path P_k;
11: compute the mapping matrix M_k^0 along the path P_k by (3) and obtain M_k^{00} by combining M_k^0 and path variable v_k as defned in (5);
12: $F \quad F + w \cdot v_k;$
13: $M^{00} \quad M^{00} + M_k^{00};$
14: end for
15: $M \quad M^{00};$
16: $T \quad T \cup \{k \mid v_{k_1} = 1; k \mid x_{k_1} = 1\};$
17: end if
18: end for
19: return T and F;

Example 3.6. Continuing to the Example 3.5, we consider encoding the objective function. For each path variable, the weight is the fdelity of that characterized path.

Thus, the objective function for the frst slice is as follows,

$$F = 0.962v_1 + 0.956v_2 + 0.988v_3 + 0.962v_4 + 0.956v_5 + 0.991v_6.$$

The solution of the frst slice is $x_1 = 0; x_2\{x_3 = 1; x_4\{x_6 = 0$. That is a SWAP gate acts on the qubits Q_6 and Q_9 inserted before the gate g_2. Similarly, the constraints and objective of each slice are encoded into a BIP problem and solved. Then the compiled quantum circuit is as shown in Fig. 4.

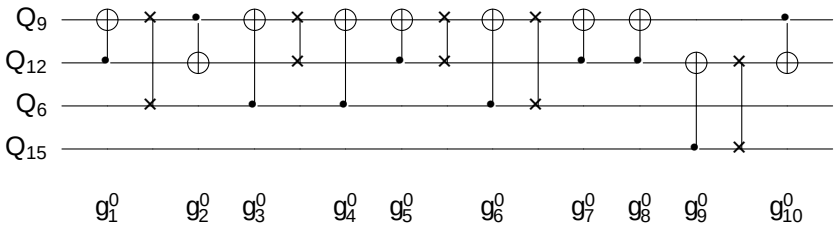


Fig. 4 The compiled quantum circuit

3.5 Complexity

The time complexity of our compilation is determined by the pruning strength. For a quantum gate g, there are $jR(g)j \times j\ Ej$ search paths. The times complexity of encoding a quantum circuit C is $jR(g)j \times j\ Ej \times j\ Cj$, where jCj is the scale of quantum circuit C.

4 Efciency

4.1 Pruning

In our approach, the number of variables is related to the size of the shortest path set. In the worst case, there are $jR(g)j \times j\ Ej$ shortest paths for encoding a gate where $jR(g)j = jQj \times j\ Qj - 1$. First, we consider pruning the set of target qubit pairs E by isomorphism, called isomorphic pruning, since pruning the currently mapped qubits set R(g) will afect the existing variables. Some heuristics with a look-ahead mechanism consider unexecutable gates in the objective function. We observed existing similar slices in quantum circuits, which is the cornerstone of the look-ahead mechanism in our approach. The isomorphic edges between the interactions of the similar slice and the target architecture are considered as the target qubit pairs, which not only satisfes the execution of the gates in that slice but also takes the following similar slices into account, which is a look-ahead mechanism. When the gates in a slice are mapped, the gates in the next similar slice are also mapped in the maximum isomorphic subgraph, which only needs inserting a few SWAP gates. The scale of target qubit pairs of a slice reduces from $jGj \times j\ Ej$ to several maximum isomorphic subgraphs, where jGj is

the number of matched gates in a slice. Other gates still use the set E as the set of target qubit pairs and the paths $P_k \in P$ that have a low fidelity will be filtered out.

Example 4.1. In Example 3.6, the first and second slices are similar, since their interactions on the same qubit set are isomorphic. The maximum isomorphic subgraph of the first slice filtered by the maximum fidelity remains $\{(Q_9, Q_{12}), (Q_{12}, Q_{15})\}$, which is also the maximum isomorphic subgraph of second slice $\{g_4, g_5, g_6\}$. Then, the path set for the second gate is $\{Q_6 \to Q_9 \to Q_{12}\}$ and the path set for the third gate is $\{Q_6 \to Q_9\}$. After isomorphic pruning, there are only 2 path variables for encoding the first slice. Compared with the encoding method without isomorphic pruning in Example 3.6, isomorphic pruning uses 4 fewer path variables.

4.2 Optimization

The optimization of qubit mapping is in three aspects, look-ahead in the initial mapping, gate cancellation and rearrangement of 1-qubit gates. In the previous work, Li et al. [7] reverses the whole circuit to optimize the initial mapping. Here we make use of this idea for our optimization. Let $C_{1:n} = C_1 C_2 \cdots C_n$ denote the first n slices in a quantum circuit C, where C_i, $1 \le i \le n$ is a slice. After compiling the fragment $C_{1:n}$, the mapping is denoted by M', which is used as the initial mapping for compiling the inverse of the fragment $C_{1:n}^{-1}$. Finally, we get the mapping M'', which is used as a new initial mapping. Our approach looks ahead to one slice for constructing the initial mapping, i.e. $n = 2$. Furthermore, the inserted SWAP gates may produce gate cancellation with the existing CX gates in the quantum circuit. Thus we consider a simple optimization of the compiled circuit, which contains gate cancellation and commutation gates [25]. The position of the SWAP gates that are inserted affects the fidelity of the neighboring 1-qubit gates. Our approach rearranges SWAP gates and 1-qubit gates to improve the fidelity of 1-qubit gates.

5 Experimental Evaluation

5.1 Implementation and Benchmarks

QMBIP is implemented in Matlab 2019a with a solver Gurobi and Python 3.7. All the experiments are conducted on a Ubuntu machine with a 2.2GHz CPU and 64G memory. Considering the factors of solving time and local optima, we use some functions of RevLib [26] with no more than 300 2-qubit gates as benchmarks, which have also been adopted in several related work. We compare QMBIP with the latest constrained-based algorithm SATMAP [14], which is based on maximum satisfiability and the industry-recognized excellent heuristic algorithm SABRE [7]. Table 2 lists the number and time cost of benchmarks successfully solved by the approaches SABRE, SATMAP and QMBIP on three sparse architectures and a dense one. The compilation time for a single instance is within 24 hours. The data of those target architectures is reported from Qiskit[4]. Figs. 5–9 more intuitively show the performance of the three methods on the benchmark and the blue markers indicate the performance of QMBIP, the green indicates SATMAP and the red indicates SABRE.

[4] https://Qiskit.org, accessed 17 December 2023.