PAPER

A SAT Approach to the Initial Mapping Problem in SWAP Gate Insertion for Commuting Gates

Atsushi MATSUO[†,††a)], *Nonmember*, Shigeru YAMASHITA[††b)], *Senior Member, and* Daniel J. EGGER[†††c)], *Nonmember*

SUMMARY   Most quantum circuits require SWAP gate insertion to run on quantum hardware with limited qubit connectivity. A promising SWAP gate insertion method for blocks of commuting two-qubit gates is a predetermined swap strategy which applies layers of SWAP gates simultaneously executable on the coupling map. A good initial mapping for the swap strategy reduces the number of required swap gates. However, even when a circuit consists of commuting gates, e.g., as in the Quantum Approximate Optimization Algorithm (QAOA) or trotterized simulations of Ising Hamiltonians, finding a good initial mapping is a hard problem. We present a SAT-based approach to find good initial mappings for circuits with commuting gates transpiled to the hardware with swap strategies. Our method achieves a 65% reduction in gate count for random three-regular graphs with 500 nodes. In addition, we present a heuristic approach that combines the SAT formulation with a clustering algorithm to reduce large problems to a manageable size. This approach reduces the number of swap layers by 25% compared to both a trivial and random initial mapping for a random three-regular graph with 1000 nodes. Good initial mappings will therefore enable the study of quantum algorithms, such as QAOA and Ising Hamiltonian simulation applied to sparse problems, on noisy quantum hardware with several hundreds of qubits.

key words: quantum compiler, quantum circuit, SWAP strategy, initial mapping

1.   Introduction

Quantum computers offer a new paradigm to address complex problems by processing information with the laws of quantum mechanics [1], [2]. In practice, quantum computing architectures have a limited qubit connectivity which is expressed as a *coupling map*, i.e., a graph in which nodes are physical qubits and edges are hardware native two-qubit gates. For example, superconducting qubits are arranged in a two-dimensional lattice such as a grid [3] or a heavy-hex graph [4].

   Most quantum circuits require gates that are not hardware native and must be decomposed into native gates [5]. Furthermore, SWAP gates are inserted in these circuits to overcome the limited qubit connectivity [6]. These tasks are done by a *transpiler* which maps program circuits into hardware executable circuits. The transpiler must therefore (i) select the physical qubits to work with (ii) perform an *initial mapping* of program qubits to the selected physical qubits and (iii) decompose the program gates and insert SWAP gates such that all program gates are implemented by hardware native gates. On noisy hardware the physical qubits can be selected by their properties such as coherence times, gate fidelities and connectivity [7], [8]. Adding SWAP gates introduces noise making it desirable to minimize their number. This minimization, done in steps (ii) and (iii), is, however, a hard combinatorial optimization problem [9], [10]. Optimal solutions to this problem can only be obtained for small quantum circuits [6], [9]. Heuristic strategies have therefore been developed [7], [11]–[14] and some included in transpilers such as Qiskit [15] and tKet [16]. Leveraging pulse-level information also helps transpilers reduce noise [17], [18].

   The initial mapping problem is NP-complete and equivalent to a subgraph isomorphism [19]. A good initial mapping significantly reduces the gate count [6] and many heuristics have been developed to produce them. For example, by assigning program qubits to physical qubits based on connectivity properties [6], [10], [19], [20]. For two-dimensional grid coupling maps a placement strategy can be devised using how often program qubits interact [12]. The SABRE (SWAP-based BidiREctional) heuristic algorithm uses circuit reversibility to iterate the initial mapping and reduce SWAP gate count [13].

   Structured circuits made of blocks of commuting two-qubit gates, such as QAOA [21] and trotterized Ising simulation [8], are prevalent in quantum computing applications. Optimal SWAP gate insertion is still a hard problem even when exploiting commutativity [20], [22]. A promising method which performs well for dense problems is to use a predetermined *swap strategy* which applies layers of SWAP gates simultaneously executable on the coupling map [23]. To map a given circuit to the hardware with swap strategies the transpiler will, for a given initial mapping, loop through the swap layers and apply any two-qubit gate that is feasible on the current qubit configuration. As the problem density increases the initial mapping becomes less relevant. However, as we show in this work, a good initial mapping significantly reduces the number of swap layers for problems that are not fully connected. Quantum computers are soon expected to have a few hundreds of noisy qubits [24]. Testing, e.g., QAOA on such quantum computers will first be done with sparse graphs close to being hardware native due to the limited gate
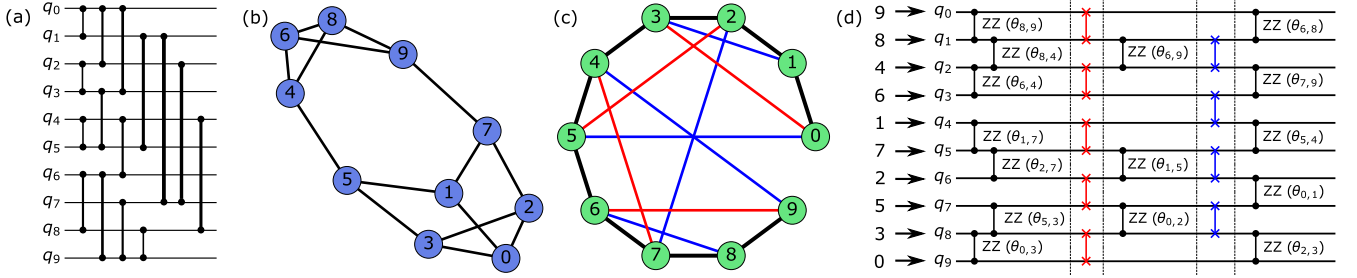
Fig. 1 Illustration of an initial mapping. (a) A quantum circuit made of commuting two-qubit ZZ gates and (b) its corresponding program graph. Each node is a program qubit and each edge a ZZ gate. (c) Connectivity graph after two layers of a line swap strategy on ten linearly connected qubits. (d) Quantum circuit of the program graph in (a) and (b) with SWAP gates inserted to match the connectivity graph in (c). The arrows indicate the initial mapping of the program qubits to physical qubits. Edge (i, j) in the program graph in (b) corresponds to a controlled phase gate with angle $\theta_{i,j}$ in (d). The layers of SWAP gates in red and blue correspond to s_1 and s_2 in a line swap strategy, respectively. The vertical barriers serve as a guide to the eye.

fidelity [23]. A procedure to generate good initial mappings for a few hundreds of qubits and sparse problems is thus needed.

Our Contribution. We propose a method to find the optimal number of swap layers even for sparse problems. First, we observe that a good initial mapping of qubits, which is not considered in [23], significantly reduces the gate count for circuits that are not fully connected. Then, we formulate this problem as a subgraph isomorphism. However, we confirmed that the resulting subgraph isomorphism problem cannot be handled by an established solver like VF2 [25] in a practical time. We thus formulate the subgraph isomorphism problem as a SAT problem to leverage powerful SAT solvers. Our method drastically reduces the gate count for a large problem with 500 qubits. Furthermore, we propose a clustering-based heuristic for even larger problems that we demonstrate with 1000 qubits in a reasonable time.

We give a brief introduction to quantum computing in Sect. 2. In Sect. 3 we describe the initial mapping problem for swap strategies which we formulate as a SAT problem in Sect. 4. We show a binary search algorithm capable of finding good initial mappings for problems with up to 500 qubits in Sect. 5. In Sect. 6 we present the heuristic algorithm that decomposes large initial mapping problems to show a gate count reduction for a 1000 qubit problem. We conclude in Sect. 7.

2. Quantum Computing

A quantum circuit, as shown in Fig. 1(a), is a model of a quantum computation. Each horizontal line is a qubit, i.e., a two-level quantum system with a state $\alpha |0\rangle + \beta |1\rangle$ such that $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$ to conserve probability. Physically, qubits are implemented by quantum systems with two controllable levels. As example, a transmon qubit is engineered by a non-linear inductor in parallel with a capacitor [26]. Each symbol between qubits is a quantum gate, i.e., a unitary matrix. Different quantum architectures implement different gate sets called the hardware native gates. For exam-

ple, fixed-frequency qubits coupled by microwave resonators implement the two-qubit CNOT gate with a cross-resonance interaction that drives a control qubit at the frequency of a target qubit [27]. A universal set of gates, such as single-qubit rotations with the CNOT gate, can build any quantum gate. Due to engineering constraints two-qubit gates cannot be applied between arbitrary qubits. This is described by a *coupling map*, i.e., a graph where each node is a physical qubit and each edge is a hardware native two-qubit gate. A quantum circuit that requires gates between non-neighboring qubits in the coupling map is *transpiled* to the hardware by inserting SWAP gates to swap program qubits.

3. Mapping Program Qubits to Physical Qubits

A swap strategy $\mathcal{S} = (S, \vec{o}, C_0)$ is a set $S = \{s_1, \ldots, s_m\}$ of m different layers of simultaneously applicable SWAP gates on a hardware coupling map C_0 and an ordering $\vec{o} = (o_1, \ldots, o_L)$ in which to apply the swap layers. Each layer of simultaneously applicable SWAP gates $s_i \in S$ is a list of pairs of physical qubits on which the SWAP gates are applied. $o_j = i \in \{1, \ldots, m\}$ implies that the jth layer of SWAP gates is $s_i \in S$. As an example, a common line swap strategy is $(\{s_1, s_2\}, (1, 2, 1, 2, \ldots), C_0)$, where $s_1 = ((0, 1), (2, 3), \ldots)$ and $s_2 = ((1, 2), (3, 4), \ldots)$. Here, we alternate swap layers on even and odd edges† of C_0 and we achieve full connectivity in $n - 2$ swap layers which is provably optimal [23]. This implies that in swap layer s_1 we apply SWAP gates to qubits 0 and 1, and qubits 2 and 3, and so on. The ordering $\vec{o}$ is $(1, 2, 1, \ldots)$, meaning that we alternate between s_1 and s_2. For a line of n physically connected qubits, the coupling map C_0 is a list of pairs $(i, i + 1)$ with $i = 0, \ldots, n - 2$. Figure 1(d) exemplifies the first two layers of a line swap strategy on 10 qubits. Here, the layers of SWAP gates in red and blue correspond to s_1 and s_2, respectively.

After l layers of the swap strategy, the hardware coupling

---

†We call an edge $(i, i + 1)$ of a linear coupling map even if $i = 0$ mod 2 and odd otherwise.
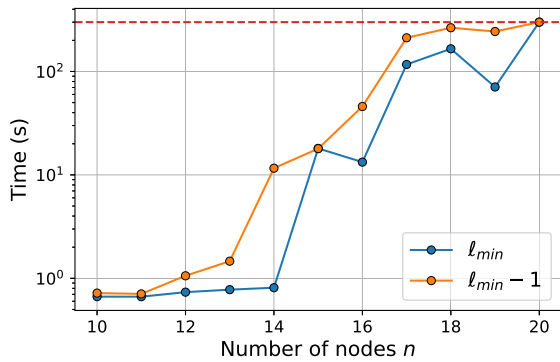
Fig. 2 Time taken by the VF2 algorithm to map a n node program graph P_n to two connectivity graphs $C_{l_{min}}$ and $C_{l_{min}-1}$. The red horizontal line is the allotted time to the VF2 algorithm.

map is transformed into the effective *connectivity graph* C_l, see Fig. 1(c). Here, the black edges indicate the original edges of the linear coupling map C_0. The red and blue edges indicate new connections introduced by the first and second layers of SWAP gates, respectively, as shown in Fig. 1(d) with matching colors. For instance, applying the first swap layer s_1 makes q_0 and q_3 adjacent since s_1 swaps qubits 0 and 1 as well as qubits 2 and 3. There is thus a red edge between qubits 0 and 3 in Fig. 1(c) because the physical qubits 1 and 2 are connected in C_0. A swap strategy achieves full connectivity if C_L is a complete graph.

We describe a block of commuting two-qubit gates by a graph $P = (V, E)$ which we call the *program graph*. Each edge corresponds to a two-qubit gate and each node is a program qubit which can be a decision variable in a QAOA. For incomplete program graphs a carefully chosen initial mapping reduces the circuit depth for a given swap strategy. An example of a random three-regular problem mapped and transpiled to a line coupling map is shown in Fig. 1. Here, S is a line swap strategy. A trivial mapping $i \rightarrow q_i$ requires a circuit with eight swap layers of S since the program qubits 0 and 2 move in the same direction in the line swap strategy. Crucially, an optimized initial mapping needs only two swap layers, see Fig. 1(d).

The initial mapping problem is a subgraph isomorphism problem where we wish to embed P in C_l. In this work we assume a given swap strategy S and seek the initial mapping $\pi : i \rightarrow q_i$ that assigns program qubit i of P to physical qubit q_i in C_0 such that the least number of swap layers is needed. I.e., find the minimum l such that C_l can embed $\pi(P)$.

We numerically investigate[†] the time taken by the VF2 solver to solve instances of the initial mapping problem with random program graphs with $n \in \{10, \ldots, 20\}$ nodes where each edge has a 20% probability of occurrence. For each n, five random graphs are mapped to the connectivity graph of a line swap strategy. We run VF2 twice: once with the smallest number of swap layers capable of embedding each graph, labeled $l_{\min}$, and once with $l_{\min} - 1$. The time taken by

---

[†]All the results presented in this work were obtained with a MacBook Pro with Apple M1 Max and 64GB memories.

VF2 grows quickly, see Fig. 2, making subgraph isomorphism solvers a poor choice to design good initial mappings for large problems. This is not surprising since the subgraph isomorphism problem is NP-complete in the general case and thus hard to solve. In a conventional subgraph isomorphism problem we seek to embed a small graph in a comparatively larger graph. However, in the initial mapping problem, the number of nodes in P and C_l are the same. Furthermore, in our case, the connectivity graphs $C_0, C_1, \ldots, C_{n-1}$ are fixed and there are known "easy" instances such as C_{n-2} (complete graph) and C_0 (line graph) for a line swap strategy. Although we do not have theoretical proofs, our experimental results indicate that finding the smallest k for C_k requires exponential time. These graphs may not have a trivial shape, and therefore, finding the smallest k may be still NP-hard in our case.

4. A SAT Formulation of the Initial Mapping

We now map the subgraph isomorphism problem in Sect. 3 to a SAT problem [28] which allows us to benefit from advances in SAT solvers [29], [30] which have also been leveraged to insert SWAP gates in quantum circuits [31], [32]. The SAT problem is expressed by literals $x_{i,j}$. Here, $x_{i,j}$ is true if node i in a program graph $P = (V, E)$ is mapped to node j in a connectivity graph $C_l = (V_l, E_l)$ which is obtained after l swap layers. There are therefore $|V||V_l|$ literals, i.e., n^2 literals if $|V| = |V_l| = n$. For example, when $x_{1,3}$ is true node 1 in the program graph is mapped to node 3 in the connectivity graph. Three conditions must be satisfied to embed the program graph into the connectivity graph C_l such that all operations in a circuit can be performed in at most l swap layers.

Condition 1. Each node i in the program graph is assigned to only one node in the connectivity graph. This condition is expressed as two sub-conditions. First, i is assigned to at least one node which implies that at least one $x_{i,j}$ is true, i.e., the logical OR of all $x_{i,j}$ for $j = 0, \ldots, |V_l| - 1$ is true

$$x_{i,0} \vee x_{i,1} \cdots \cdots \vee x_{i,|V_l|-1} = 1. \tag{1}$$

Second, i is assigned to at most one node in the connectivity graph. For example, we do not want $x_{i,j}$ and $x_{i,k}$ with $j \neq k$ to be simultaneously true, i.e.,

$$\neg x_{i,j} \vee \neg x_{i,k} = 1 \quad \text{for} \quad j \neq k. \tag{2}$$

Therefore, to assign node i to at most one node in the connectivity graph the clause in Eq. (2) must be true for all pairs of nodes in V_l, i.e.,

$$\bigwedge_{j > k} \neg x_{i,j} \vee \neg x_{i,k} = 1. \tag{3}$$

If Eqs. (1) and (3) hold for all i then condition 1 is true. Condition 1 thus generates $|V|$ clause of length $|V_l|$ and $|V||V_l|(|V_l| - 1)/2$ clauses of length 2.

Condition 2. At most one node in a program graph is assigned to a node in a connectivity graph. For node $k \in V_l$

the condition $\neg x_{i,k} \vee \neg x_{j,k} = 1$ prohibits simultaneously assigning nodes i and j to k. The conjunction of such clauses over all $i \neq j$ prevents assigning multiple nodes of P to a node k in C_l

$$\bigwedge_{i>j} \neg x_{i,k} \vee \neg x_{j,k} = 1. \qquad (4)$$

Condition 2 thus creates $|V_l||V|(|V|-1)/2$ clauses of length 2.

Condition 3. Adjacent nodes in a program graph must be adjacent after they are mapped to nodes in the connectivity graph. Accordingly, for an edge $(i,j) \in E$ then $x_{i,k}$ being true implies that there must be a $x_{j,k'}$ with $(k,k') \in E_l$ that is also true. This implication is expressed as

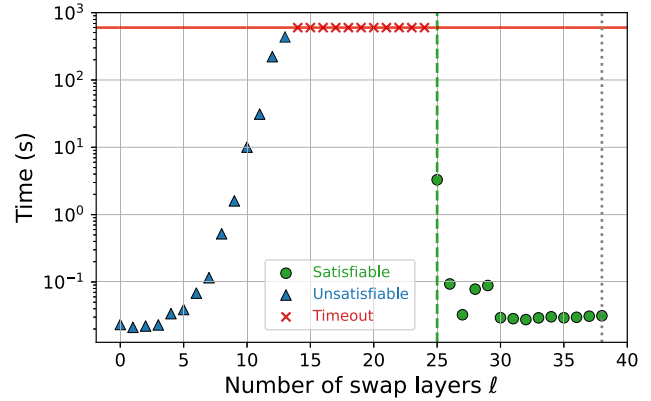$$\neg x_{i,k} \vee \left(\bigvee_{(k,k') \in E_l} x_{j,k'} \right). \qquad (5)$$

We can thus express condition 3 by taking the conjunction of all clauses (5) generated by each edge $(i,j) \in E$. Condition 3 thus generates $|E||V_l|$ clauses of variable length.

The conjunction of the clauses of Conditions 1, 2 and 3 yields a SAT formulation of the initial mapping problem. A SAT solver can therefore determine if there exists an initial qubit placement such that all gates in the circuit can be implemented by at most l swap layers of the swap strategy. If such a placement exists the SAT solver also returns a satisfying variable assignment.

5. Finding the Best Initial Mapping Based on the SAT Formulation

We solve the initial mapping problem with PySAT [30], a Python library designed to solve SAT problems. PySAT provides built-in SAT solvers with low-level implementations such as C++, making it fast and easy to use. In the experiments, we use a linear coupling map and the line swap strategy described in Sect. 3 to represent the limited qubit connectivity in current quantum devices.

As a preliminary experiment, we solve the initial mapping problem for a random graph with 40 nodes, where each edge has a 20% probability of occurrence. We chose a sparse graph since swap strategies without an initial mapping do not perform well on them [23]. We allow PySAT 600 seconds[†] to determine if an instance is satisfiable. Since we transpile the program graph to a linear coupling map we solve 39 SAT instances; one for each $l \in \{0, \ldots, 38\}$. The SAT instances with a number of swap layers $l < 14$ are not satisfiable, see Fig. 3, i.e., more swap layers are needed to overcome the limited qubit connectivity. The SAT instances with $l > 25$ are satisfiable and the circuit can be executed on the hardware. PySAT cannot determine if the problem is satisfiable in 10 minutes for $l \in \{14, \ldots, 24\}$. We therefore observe the

---

[†]We chose 600 seconds to make the optimization time manageable on a laptop. In a high-performance computing environment more resources may be allotted to the SAT solver.

Fig. 3 Time taken by PySAT to determine if a 40 node graph can be embedded in a connectivity graph as a function of the number of swap layers. The red horizontal line indicates the allotted time to the SAT solver. The green dashed vertical line shows the resulting l. The grey dotted line shows the number of swap layers required by a trivial mapping.

typical easy-hard-easy pattern of SAT instances [33], [34]. Crucially, the satisfiable SAT instance at $l = 25$ reduces the number of swap layers by 34% compared to a trivial initial mapping. Note that the specific numbers 14 and 25 depend on the graph instance, but the easy-hard-easy pattern does not.

Based on these observations we can find a good initial mapping to reduce the number of swap layers with a binary search over l. Since the number of swap layers grows linearly with problem size [23] we need only solve $O(\log n)$ SAT instances. The initial points of the binary search are $l_L = 0$ and $l_R = |V| - 2$, which are typically satisfiable and unsatisfiable, respectively, and easy to solve, e.g., see Fig. 3. Furthermore, we allow the SAT solver a fixed time to determine if a SAT instance is satisfiable. If the SAT solver cannot find a solution in the allowed time we consider the SAT instance to be unsatisfiable. If $\mathcal{S}$ can reach full connectivity we are guaranteed to find an initial mapping in the worst case with an l equivalent to full connectivity. Once the binary search has converged we make an initial mapping based on the solution of the last satisfiable SAT instance. The algorithm is summarized below.

We test Algorithm 1 on two types of sparse random graphs commonly used as benchmarks [23], [35], [36]: random three-regular graphs ($RR3_n$) and random graphs in which edges appear with a 20% probability ($Rand_n$). We conduct experiments on graphs with $n = 40, 100, 200$, and 500 nodes and linear coupling maps. It is worth noting that Algorithm 1 works with any coupling map and swap strategy. For each graph size n, we perform experiments on five different graph instances. We compare Algorithm 1 to a random initial mapping and a trivial mapping, which maps program qubit v_i in a program graph to physical qubit v_i' in a connectivity graph. We average over 100 different random initial mappings for each graph and chose the result with the minimum number of swap layers.

We allow PySAT a maximum of 600 seconds per SAT instance $SAT_{embed}(P, C_l)$. We observe a significant reduction

Algorithm 1 The algorithm to find a good initial qubit mapping.

Input: program graph $P = (V, E)$, swap strategy $\mathcal{S} = (S, \vec{o}, C_0)$
Output: Initial layout that reduces the number of swap layers of $\mathcal{S}$ needed to execute P on the hardware.
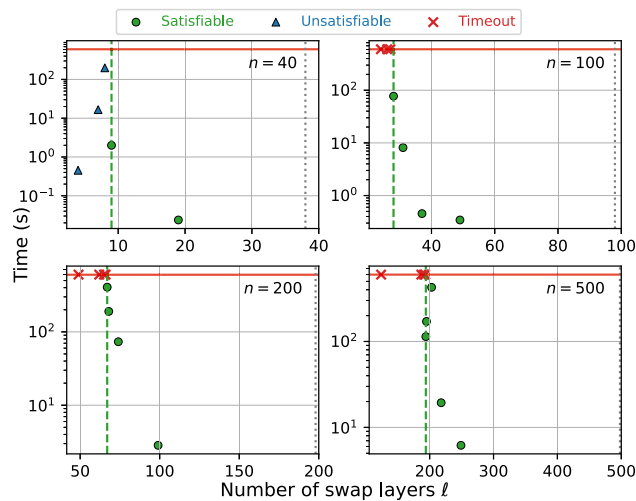1: $l_L \leftarrow 0$
2: $l_R \leftarrow |V| - 2$
3: **while** $l_L < l_R$ **do**
4:     $l \leftarrow \lfloor (l_L + l_R)/2 \rfloor$
5:     Create connectivity graph C_l from $\mathcal{S}$
6:     Create SAT instance $\text{SAT}_{\text{embed}}(P, C_l)$
7:     **if** $\text{SAT}_{\text{embed}}(P, C_l)$ is satisfiable **then**
8:         $l_R \leftarrow l$
9:     **else**
10:        $l_L \leftarrow l + 1$
11:     **end if**
12: **end while**

Fig. 4 Time taken of each iteration of the binary search to map three-regular graphs with a different number of nodes n. The red horizontal line indicates the allotted time to the SAT solver. The green dashed vertical line shows the resulting l_{min}. The grey dotted line shows the number of swap layers required by a trivial mapping.

in the number of SWAP layers needed, see Fig. 4. The optimal l may exist in the SAT instances which timed-out. Crucially, it is possible to find a good yet suboptimal l that significantly decreases the number of swap layers, and the number of CNOT gates in a practical time. For example, Algorithm 1 identifies an initial mapping for a 500 node three-regular graph that reduces the number of swap layers to less than 200 in 90 minutes while a trivial mapping requires full connectivity. The SAT approach significantly decreases the number of CNOT gates and circuit depth as shown in Table 1 which lists the number of swap layers, and the number of the CNOT gates averaged over the five graph instances at each size. The trivial mapping nearly always requires full connectivity, i.e., $l = n - 2$ swap layers. Random initial mappings only make minor improvements although the best result was chosen from 100 different trials. We do not expect this situation to change with a polynomial increase in the number of trials since the search space scales combinatorially

with n.

We also apply the SABRE layout, implemented in Qiskit [15], to find an initial mapping before inserting swap gates with a swap strategy. SABRE layout finds an initial mapping with an iterative bidirectional routing of the program circuit [13] for general swap insertion methods. Since the SABRE layout method is not tailored to the swap strategy its results are as bad as the trivial mapping and the random mapping, see Table 1. This result emphasizes the need for an initial mapping method tailored to the swap strategy. We also compare the SWAP gate depth of a swap strategy with an initial mapping to SABRE swap routing [13], a heuristic swap insertion algorithm, applied after the SABRE layout. The numbers of swap layers of Sabre swap routing, counted as the SWAP gate depth of the resulting quantum circuit, is much larger than the swap strategy since SABRE swap routing does not fully utilize the ZZ gate commutativity. We note that each layer of SWAP gates inserted by SABRE swap routing is less dense than in a swap strategy. However, circuit depth is crucial for current noisy devices since their coherence time is limited.

The reduction from the SAT approach is especially large for random three-regular graphs since they are sparser than random graphs. These results highlight that a good initial mapping significantly reduces the number of swap layers. Furthermore, since the number of SWAP gates in each layer of a swap strategy linearly increases with the number of qubits n, decreasing the number of swap layers decrease the total number of CNOT gates more in the larger program graphs.

6. Additional Strategies for Scalability

Algorithm 1 presented in Sect. 5 constructs a SAT problem with n^2 variables. Furthermore, we could not run the 500 nodes random graph instances on the MacBook as they required more than 64 GB of memory and a single iteration of the binary search took more than 20 minutes. To make the problem more manageable we introduce a heuristic to perform the initial mapping that iteratively maps sub-graphs of the program graph to the connectivity graph. More formally, for a program graph $P = (V, E)$ with n variables and a corresponding swap strategy $\mathcal{S}$ we select a sub-graph $P_0 = (V_0, E_0) \subset P$ such that $|V_0| = n_0$. We then find the l_0 with Algorithm 1 that allows us to embed P_0 in C_{l_0}. The resulting SAT problem has $n_0 n$ decision variables with $n_0 < n$. We iterate the mapping. At iteration i we select a subgraph $P_i = (V_i, E_i) \subset P$ such that $P_i \cap (\cup_{j=0}^{i-1} P_j) = \emptyset$ and build the SAT problem such that the nodes of P_i are mapped to unassigned qubits with the condition that any edge connecting a node from V_i to $\cup_{j=0}^{i-1} V_j$ also has an edge in the connectivity graph C_{l_i} of $\mathcal{S}$. To find l_i we perform a binary search in the interval $\{l_{i-1}, \ldots, L\}$ with $L = n - 2$ for a line swap strategy.

We test this heuristic on a random three-regular graph with 1000 nodes. The sub-graphs P_i are found with the spectral clustering [37] implemented in Scikit-learn [38]. We test the heuristic twice: once with ten clusters of 100 nodes and once with five clusters of 200 nodes. Under these

Table 1 Number of swap layers and CNOT gates needed to execute a program graph on the hardware following different initial mappings obtained with the trivial, random, SABRE layout, and SAT strategies. The "Sabre layout" column is the number of layers of the line swap strategy after an initial mapping found by SABRE layout. The "SABRE swap" column shows the swap depth of a circuit after SABRE layout and SABRE swap routing. Each swap layer has, up to edge effects, $n/2$ SWAP gates and each SWAP gate requires three CNOT gates. η is the ratio of the number of CNOT gates found with the SAT solver to the random initial mapping.

| | Number of swap layers | | | | | Number of CNOT gates | | | |
Graph	Trivial	Random	SABRE layout	SABRE swap	SAT	Trivial	Random	SAT	η
$RR3_{40}$	38 ± 0	35 ± 1	38 ± 1	52 ± 12	9 ± 0	2212 ± 23	2060 ± 44	516 ± 24	0.25
$Rand_{40}$	38 ± 0	38 ± 0	38 ± 0	181 ± 14	25 ± 1	2223 ± 0	2200 ± 28	1454 ± 79	0.66
$RR3_{100}$	98 ± 1	95 ± 1	97 ± 1	224 ± 54	27 ± 1	14494 ± 119	14049 ± 201	4040 ± 174	0.29
$Rand_{100}$	98 ± 0	98 ± 0	98 ± 0	921 ± 145	84 ± 1	14553 ± 0	14553 ± 0	12475 ± 94	0.85
$RR3_{200}$	198 ± 0	195 ± 1	198 ± 0	680 ± 66	68 ± 2	59103 ± 0	58149 ± 120	20239 ± 513	0.35
$Rand_{200}$	198 ± 0	198 ± 0	198 ± 0	2322 ± 313	183 ± 0	59103 ± 0	59103 ± 0	54746 ± 146	0.93
$RR3_{500}$	498 ± 0	495 ± 1	498 ± 0	4373 ± 285	196 ± 6	372454 ± 366	370358 ± 872	147006 ± 4405	0.35
$Rand_{500}$	498 ± 0	498 ± 0	498 ± 0	18290 ± 14119	$-$	372753 ± 0	372753 ± 0	$-$	$-$
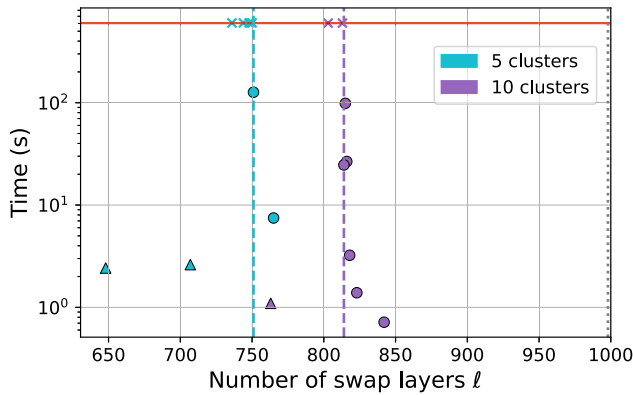
Fig. 5 Time taken of each iteration of the binary search to map a three-regular graph with 1000 nodes to a linear coupling map. Blue and purple markers correspond to data in which the sub-graphs P_i have 200 and 100 nodes, respectively. The circle, triangle, and cross markers correspond to satisfiable, unsatisfiable, and timeout, respectively. The red horizontal line indicates the allotted time to the SAT solver. The blue and purple dashed vertical lines show the resulting l_{min} for the 200 and 100 node clustering, respectively. The grey dotted line shows the number of swap layers required by a trivial mapping.

conditions the problem is manageable on a MacBook Pro and we identify an initial mapping that reduces the number of swap layers, see Fig. 5. With sub-graphs P_i with 200 and 100 nodes we find a $l_{\min} = 751$ and $l_{\min} = 814$, respectively. Smaller sub-graphs are computationally easier to manage but produce initial mappings with a larger number of swap layers. This is expected since more clusters simplify the problem.

7. Conclusion

Swap strategies can efficiently transpile circuits made of blocks of commuting two-qubit gates to hardware resulting in low-depth circuits [23]. However, a good initial mapping further reduces the number of required swap gates for program graphs that are not complete. When formulated as a subgraph isomorphism the initial mapping problem of embedding a program graph P in a connectivity graph C_l can only be solved for small instances with less than ~ 20 nodes.

We therefore developed a SAT-based approach that finds good initial mappings for circuits with commuting gates that are transpiled to the hardware with swap strategies. We formulated the subgraph isomorphism as a SAT instance to benefit from the progress in SAT solvers. A binary search reduces the number of swap layers by finding an initial mapping in $O(\log n)$ steps. We also proposed a heuristic approach to map graphs that are too large to be handled as a single SAT instance. The heuristic approach divides the program graph into several clusters, and iteratively applies the binary search over the resulting smaller SAT instances. Our results show a significant decrease in the number of swap layers and CNOT gates for program graphs with up to 1000 qubits.

Future work may devise more efficient heuristics to solve the initial mapping problem on large instances. Nevertheless, the methodology proposed here will allow us to map, e.g., sparse QAOA circuits to the quantum hardware to be developed in the coming years [24]. The initial mapping will be crucial since the gate fidelity limits the structure of the program graph to sparse graphs that reassembles the coupling map [23], [39].

References

[1] P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," Proc. 35th Annual Symposium on Foundations of Computer Science, pp.124–134, 1994.

[2] N. Moll, P. Barkoutsos, L.S. Bishop, J.M. Chow, A. Cross, D.J. Egger, S. Filipp, A. Fuhrer, J.M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli, and K. Temme, "Quantum optimization using variational algorithms on near-term quantum devices," Quantum Sci. Technol., vol.3, no.3, p.030503, 2018.

[3] M.P. Harrigan, K.J. Sung, M. Neeley, K.J. Satzinger, F. Arute, K. Arya, J. Atalaya, J.C. Bardin, R. Barends, S. Boixo, M. Broughton, B.B. Buckley, D.A. Buell, B. Burkett, N. Bushnell, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, S. Demura, A. Dunsworth, D. Eppens, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Graff, S. Habegger, A. Ho, S. Hong, T. Huang, L.B. Ioffe, S.V. Isakov, E. Jeffrey, Z. Jiang, C. Jones, D. Kafri, K. Kechedzhi, J. Kelly, S. Kim, P.V. Klimov, A.N. Korotkov, F. Kostritsa, D. Landhuis, P. Laptev, M. Lindmark, M. Leib, O. Martin, J.M. Martinis, J.R. McClean,

M. McEwen, A. Megrant, X. Mi, M. Mohseni, W. Mruczkiewicz, J. Mutus, O. Naaman, C. Neill, F. Neukart, M.Y. Niu, T.E. O'Brien, B. O'Gorman, E. Ostby, A. Petukhov, H. Putterman, C. Quintana, P. Roushan, N.C. Rubin, D. Sank, A. Skolik, V. Smelyanskiy, D. Strain, M. Streif, M. Szalay, A. Vainsencher, T. White, Z.J. Yao, P. Yeh, A. Zalcman, L. Zhou, H. Neven, D. Bacon, E. Lucero, E. Farhi, and R. Babbush, "Quantum approximate optimization of non-planar graph problems on a planar superconducting processor," Nat. Phys., vol.17, no.3, pp.332–336, 2021.

[4] C. Chamberland, G. Zhu, T.J. Yoder, J.B. Hertzberg, and A.W. Cross, "Topological and subsystem codes on low-degree graphs with flag qubits," Phys. Rev. X, vol.10, no.1, p.011022, 2020.

[5] K. Iwama, Y. Kambayashi, and S. Yamashita, "Transformation rules for designing CNOT-based quantum circuits," Proc. 39th Annual Design Automation Conference, DAC'02, New York, NY, USA, pp.419–424, Association for Computing Machinery, 2002.

[6] M.Y. Siraichi, V.F. dos Santos, C. Collange, and F.M.Q. Pereira, "Qubit allocation," Proc. 2018 International Symposium on Code Generation and Optimization, CGO 2018, New York, NY, USA, p.113–125, Association for Computing Machinery, 2018.

[7] S.S. Tannu and M.K. Qureshi, "Not all qubits are created equal: A case for variability-aware policies for NISQ-era quantum computers," Proc. 24th Int. Conf. on Architectural Support for Program. Languages and Oper. Syst. (ASPLOS), New York, NY, USA, pp.987–999, Association for Computing Machinery, 2019.

[8] A.C. Vazquez, D.J. Egger, D. Ochsner, and S. Woerner, "Well-conditioned multi-product formulas for hardware-friendly hamiltonian simulation," Quantum, vol.7, p.1067, 2023.

[9] A. Lye, R. Wille, and R. Drechsler, "Determining the minimal number of SWAP gates for multi-dimensional nearest neighbor quantum circuits," The 20th Asia and South Pacific Design Automation Conference, pp.178–183, 2015.

[10] P. Murali, A. Javadi-Abhari, F.T. Chong, and M. Martonosi, "Formal constraint-based compilation for noisy intermediate-scale quantum systems," Microprocess. and Microsys., vol.66, pp.102–112, 2019.

[11] A. Kole, K. Datta, and I. Sengupta, "A heuristic for linear nearest neighbor realization of quantum circuits by SWAP gate insertion using N-gate lookahead," IEEE Trans. Emerg. Sel. Topics Circuits Syst., vol.6, no.1, pp.62–72, 2016.

[12] A. Bhattacharjee, C. Bandyopadhyay, R. Wille, R. Drechsler, and H. Rahaman, "A novel approach for nearest neighbor realization of 2D quantum circuits," 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp.305–310, 2018.

[13] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for NISQ-era quantum devices," Proc. 24th Int. Conf. on Architectural Support for Program. Languages and Oper. Syst. (ASPLOS), New York, NY, USA, pp.1001–1014, Association for Computing Machinery, 2019.

[14] T. Itoko, R. Raymond, T. Imamichi, and A. Matsuo, "Optimization of quantum circuit mapping using gate transformation and commutation," Integration, vol.70, pp.43–50, 2020.

[15] M.S. Anis, A. Mitchell, H. Abraham, A. Offei, R. Agarwal, G. Agliardi, et al., "Qiskit: An open-source framework for quantum computing," 2021.

[16] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "t|ket⟩: A retargetable compiler for NISQ devices," Quantum Sci. Technol., vol.6, no.1, p.014003, 2020.

[17] T. Alexander, N. Kanazawa, D.J. Egger, L. Capelluto, C.J. Wood, A. Javadi-Abhari, and D.C. McKay, "Qiskit pulse: Programming quantum computers through the cloud with pulses," Quantum Sci. Technol., vol.5, no.4, p.044006, 2020.

[18] N. Earnest, C. Tornow, and D.J. Egger, "Pulse-efficient circuit transpilation for quantum applications on cross-resonance-based hardware," Phys. Rev. Research, vol.3, no.4, p.043088, 2021.

[19] D. Maslov, S.M. Falconer, and M. Mosca, "Quantum circuit placement," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.27, no.4, pp.752–763, 2008.

[20] M. Alam, A. Ash-Saki, and S. Ghosh, "Circuit compilation methodologies for quantum approximate optimization algorithm," 53rd Annual IEEE/ACM Int. Symp. on Microarch. (MICRO), pp.215–228, 2020.

[21] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014.

[22] L. Lao and D.E. Browne, "2QAN: A quantum compiler for 2-local qubit Hamiltonian simulation algorithms," Proc. 49th Annual Int. Symp. Comput. Architecture, ISCA'22, New York, NY, USA, p.351–365, Association for Computing Machinery, 2022.

[23] J. Weidenfeller, L.C. Valor, J. Gacon, C. Tornow, L. Bello, S. Woerner, and D.J. Egger, "Scaling of the quantum approximate optimization algorithm on superconducting qubit based hardware," Quantum, vol.6, p.870, 2022.

[24] J. Gambetta, "Expanding the IBM Quantum roadmap to anticipate the future of quantum-centric supercomputing," 2022.

[25] L. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," IEEE Trans. Pattern Anal. Mach. Intell., vol.26, no.10, pp.1367–1372, 2004.

[26] J. Koch, T.M. Yu, J. Gambetta, A.A. Houck, D.I. Schuster, J. Majer, A. Blais, M.H. Devoret, S.M. Girvin, and R.J. Schoelkopf, "Charge-insensitive qubit design derived from the cooper pair box," Phys. Rev. A, vol.76, no.4, p.042319, 2007.

[27] S. Sheldon, E. Magesan, J.M. Chow, and J.M. Gambetta, "Procedure for systematically tuning up cross-talk in the cross-resonance gate," Phys. Rev. A, vol.93, no.6, p.060302, 2016.

[28] J. Torán, "On the resolution complexity of graph non-isomorphism," Theory and Applications of Satisfiability Testing – SAT 2013, M. Järvisalo and A. Van Gelder, eds., Berlin, Heidelberg, pp.52–66, Springer Berlin Heidelberg, 2013.

[29] T. Balyo, N. Froleyks, M. Heule, M. Iser, M. Järvisalo, and M. Suda, "Proceedings of SAT Competition 2021: Solver and Benchmark Descriptions," 2021.

[30] A. Ignatiev, A. Morgado, and J. Marques-Silva, "PySAT: A Python toolkit for prototyping with SAT oracles," SAT, pp.428–437, 2018.

[31] W. Hattori and S. Yamashita, "Quantum circuit optimization by changing the gate order for 2D nearest neighbor architectures," Reversible Computation, J. Kari and I. Ulidowski, eds., pp.228–243, Springer International Publishing, Cham, 2018.

[32] A. Matsuo, W. Hattori, and S. Yamashita, "Reducing the overhead of mapping quantum circuits to IBM Q system," IEEE Int. Symp. Circuits Syst. (ISCAS), pp.1–5, 2019.

[33] I.P. Gent and T. Walsh, "The SAT phase transition," Proc. 11th European Conference on Artificial Intelligence, ECAI'94, USA, pp.105–109, John Wiley & Sons, 1994.

[34] C. McCreesh, P. Prosser, C. Solnon, and J. Trimble, "When subgraph isomorphism is really hard, and why this matters for graph databases," J. Artif. Int. Res., vol.61, no.1, p.723–759, 2018.

[35] S. Bravyi, A. Kliesch, R. Koenig, and E. Tang, "Obstacles to variational quantum optimization from symmetry protection," Phys. Rev. Lett., vol.125, no.26, p.260505, 2020.

[36] L. Zhou, S.T. Wang, S. Choi, H. Pichler, and M.D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," Phys. Rev. X, vol.10, no.2, p.021067, 2020.

[37] J. Shi and J. Malik, "Normalized cuts and image segmentation," IEEE Trans. Pattern Anal. Mach. Intell., vol.22, no.8, pp.888–905, 2000.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol.12, no.85, pp.2825–2830, 2011.

[39] D.S. França and R. García-Patrón, "Limitations of optimization algorithms on noisy quantum devices," Nat. Phys., vol.17, no.11, pp.1221–1227, 2021.

Atsushi Matsuo received his B.E. and M.E. degrees in Information Science and Engineering from Ritsumeikan University, Shiga, Japan in 2011 and 2013, respectively. He is currently a Ph.D. candidate at the Graduate School of Information Science and Engineering, Ritsumeikan University, while working at IBM Research – Tokyo. His research interests include quantum circuit synthesis and variational quantum algorithms.

Shigeru Yamashita is a professor at the Department of Computer Science, College of Information Science and Engineering, Ritsumeikan University. He received his B.E., M.E. and Ph.D. degrees in Information Science from Kyoto University, Kyoto, Japan, in 1993, 1995 and 2001, respectively. His research interests include new types of computation and logic synthesis for them. He received the 2000 IEEE Circuits and Systems Society Transactions on Computer-Aided Design of Integrated Circuits and Systems Best Paper Award, SASIMI 2010 Best Paper Award, 2010 IPSJ Yamashita SIG Research Award, and 2010 Marubun Academic Achievement Award of the Marubun Research Promotion Foundation. He is a senior member of IEEE, and a member of IPSJ.

Daniel J. Egger received the B.Sc. and M.Sc. degrees in physics from the Ecole Polytechnique Federal de Lausanne in 2008 and 2011, respectively. He obtained a Ph.D. in physics from the University of Saarland in 2014 with distinction. From 2014 to 2016 he was a Risk Manager at Partners Group in Zug and has since been working for IBM Research – Zürich GmBH. His research interests include the applications of quantum algorithms and the control of quantum computers at the level of control pulses.