On Optimal Subarchitectures for Quantum Circuit Mapping

TOM PEHAM, Technical University of Munich, Germany
LUKAS BURGHOLZER, Johannes Kepler University, Austria
ROBERT WILLE, Technical University of Munich, Germany and Software Competence Center Hagenberg GmbH, Austria

Compiling a high-level quantum circuit down to a low-level description that can be executed on state-of-the-art quantum computers is a crucial part of the software stack for quantum computing. One step in compiling a quantum circuit to some device is quantum circuit mapping, where the circuit is transformed such that it complies with the architecture's limited qubit connectivity. Because the search space in quantum circuit mapping grows exponentially in the number of qubits, it is desirable to consider as few of the device's physical qubits as possible in the process. Previous work conjectured that it suffices to consider only subarchitectures of a quantum computer composed of as many qubits as used in the circuit. In this work, we refute this conjecture and establish criteria for judging whether considering larger parts of the architecture might yield better solutions to the mapping problem. We show that determining subarchitectures that are of minimal size, i.e., from which no physical qubit can be removed without losing the optimal mapping solution for some quantum circuit, is a very hard problem. Based on a relaxation of the criteria for optimality, we introduce a relaxed consideration that still maintains optimality for practically relevant quantum circuits. Eventually, this results in two methods for computing near-optimal sets of subarchitectures—providing the basis for *efficient* quantum circuit mapping solutions. We demonstrate the benefits of this novel method for state-of-the-art quantum computers by IBM, Google and Rigetti.

1 INTRODUCTION

Quantum computing [1] is an emerging technology where computations are governed by quantum-mechanical principles. Despite its relative infancy, even currently available quantum computers impose many challenges on quantum algorithm designers—increasing the need for design automation methods in the realm of quantum computing. One critical challenge (for devices based on superconducting qubits [2]) arises from the limited connectivity of the physical qubits on these quantum computers. Instead of allowing multi-qubit gates between arbitrary qubits, a *coupling map* dictates which pairs of qubits may interact with each other over the course of a computation. Any quantum circuit executed on an actual device needs to conform to those restrictions. Thus, similar to classical computing, high-level quantum algorithms need to be *compiled* to the target architecture—encompassing many individual steps, such as synthesis, qubit allocation/placement, qubit routing, optimization, and scheduling [3]–[14].

In the following, we focus on the steps dictated by the limited connectivity of the devices, i.e., qubit allocation/placement and routing, which is commonly referred to as *quantum circuit mapping*. Mapping a quantum circuit first requires *allocating* the circuit's logical qubits on the device's physical qubits, i.e., identifying a subarchitecture of the whole device where the circuit shall be executed. Then, the logical qubits need to be *routed* on the physical qubits so that any operation in the circuit is executed on qubits that are connected on the device's coupling map. This is commonly conducted by inserting SWAP gates into the circuit which allow dynamically changing the logical-to-physical qubit assignment [3]–[9], [12]–[14].

In order to ensure reliable execution of the resulting circuit, it is crucial to keep the overhead introduced through routing as small as possible. Unfortunately, determining optimal mappings of quantum circuits is an NP-hard problem [7], [15]—mainly due to the involved search space growing exponentially in the number of considered qubits. This is exacerbated by the fact that quantum computers are not available in arbitrary sizes. At the time of writing, e.g., IBM hosts systems using 1, 5, 7, 27, 65, 127, and 433 qubits. This means that, in many cases, significantly more qubits are available on an architecture than are strictly needed for mapping a quantum circuit. This

is going to become even more problematic as the size-gap between newer generations of quantum computers increases. For example, IBM plans to provide a quantum computer with 1121 qubits in 2023 [16]. A quick solution for this problem is to just pick a subarchitecture of the device that contains as many qubits as needed. While this keeps the search space for qubit routing as small as possible, it could be possible that incorporating more qubits than needed—effectively increasing the search space—allows for solutions requiring less overhead. Experimental results on optimal quantum circuit mapping conjectured that this is not the case [13], [14]—suggesting that optimality is preserved by restricting the search space to the bare minimum.

In this work, we answer this question by showing that considering larger subarchitectures can indeed yield better mappings. Naturally, the next step is to ask which subarchitectures *should* be considered during mapping so that no essential parts of the search space are cut off. Unsurprisingly, determining subarchitectures that are as small and that contain *all* optimal mapping solutions for any quantum circuit of a specified size turns out to be a very hard problem. Luckily, the general notion of such *optimal subarchitectures* is hardly needed for mapping practical quantum circuits. This work defines and analyzes minimal sets of subarchitectures that allow for optimal mapping solutions for all but the most esoteric quantum circuits with a certain number of qubits, while drastically reducing the search space in many cases. To this end, the subarchitectures are ranked according to their coverage of optimality, i.e., the number of quantum circuits that can be optimally mapped to a given subarchitecture.

The resulting software is integrated into the open-source tool QMAP (available at https://github.com/cda-tum/qmap), which is part of the *Munich Quantum Toolkit* (MQT), and can be used to compute various sets of subarchitectures with a trade-off between complexity and coverage of optimality. For convenience, it includes a pre-computed library of subarchitectures for common quantum computing devices. Our results considering state-of-the-art quantum computers demonstrate the benefits of using the generated sets as the basis for quantum circuit mappers—allowing to significantly reduce the overall complexity of the search space—while still ensuring that essential parts of the search space are covered for most quantum circuits.

The rest of this paper is structured as follows. Section 2 provides the necessary background in graph theory and quantum circuit mapping required to present the ideas in this work. After Section 3 introduces the problem this work aims to address, Section 4 defines two criteria that allow for the existence of better mapping solutions on larger subarchitectures and gives a complete characterization of optimal subarchitectures. Two algorithms for constructing near-optimal subarchitectures are presented in Section 5. Section 6 demonstrates the benefits of this novel method for existing quantum computing architectures. Finally, Section 7 concludes this paper.

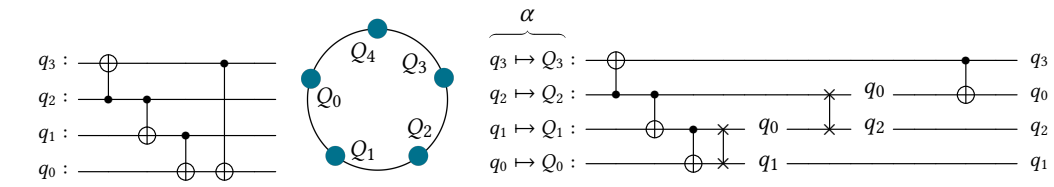
2 BACKGROUND

The connectivity of qubits in modern NISQ [17] devices is described via the architecture's coupling graph. Therefore, a lot of jargon related to graph theory will be used in the discussions on optimal subarchitectures. This section briefly introduces the relevant graph-theoretic concepts as well as the basics of quantum circuit mapping. For a comprehensive introduction to graph theory the reader is referred to any standard textbook such as [18].

2.1 Graph Theory

A *graph* $A = (V, E)$ is comprised of a set of *vertices* V and a set of *edges* $E \subseteq V \times V$ connecting vertices. The size of a graph is the number of vertices in the graph denoted $|A| = |V|$. A is called *undirected* if the pair $(v, w) \in E$ is unordered, otherwise it is called directed.

Two graphs $A = (V_A, E_A)$, $A' = (V_{A'}, E_{A'})$ are *isomorphic* (denoted $A \cong A'$) if there is a bijective function $h : V_A \longrightarrow V_{A'}$ such that $(h(v), h(w)) \in E_{A'}$ iff $(v, w) \in E_A$.



(a) Quantum Circuit (b) Architecture (c) Mapped circuit

Fig. 1. Example instance of the mapping problem

A graph $A' = (V_{A'}, E_{A'})$ is a *subgraph* of $A = (V_A, E_A)$ (denoted $A' \sqsubseteq A$) if $V_{A'} \subseteq V_A$ and $E_{A'} \subseteq E_A$. If A' is a proper subgraph, i.e., $A' \neq A$ we write $A' \sqsubset A$. A' is *subgraph isomorphic* to A if there is a subgraph $A'' \sqsubseteq A$ such that $A' \cong A''$. An *induced subgraph* of a vertex set $V' \subseteq V$ in $A = (V, E)$ is a subgraph $A' \sqsubseteq A$ with $A' = (V', E')$ and the property that, if any vertices $v, w \in V'$ are connected in A, then $(v, w) \in E'$. In an overloading of notation we also call a subgraph $A' = (V', E')$ of A an induced subgraph, if A' is the induced subgraph of V' in A.

A *path* in a graph $A = (V, E)$ is a (non-empty) sequence of distinct vertices $p = (v_0, v_1, \cdots, v_{n-1})$ with $v_i \in V, 0 \leq i < n$ and $(v_i, v_{i+1}) \in E, 0 \leq i < n$. The length of this path is denoted $|p|$. For two vertices $v, w \in V$, $p_A(v, w)$ gives a shortest path between v and w in A. The longest shortest path in a graph is called the graph's *diameter*.

2.2 Quantum Circuit Mapping

Although the topic of quantum computing is vast and fascinating, understanding its intricacies is not required in order to understand the contents of this work. All one needs to know about quantum computing is that, as in classical computing, there is a circuit model for quantum computing. Graphically, a quantum circuit is made up of *wires* representing qubits and *gates* representing transformations of qubits denoted on the respective wires. A quantum circuit can then be written as a sequence of gates $G = g_0 \cdots g_{|G|-1}$, where $|G|$ denotes the number of gates in the circuit (also called the circuit's size). Quantum gates can act on individual qubits or on multiple qubits simultaneously.

Existing quantum computers usually offer a native gate set comprised of a family of singe-qubit gates complemented with some two-qubit gate (e.g. the controlled not or CNOT gate). In addition, these quantum computers often do not allow execution of two-qubit gates between any qubit pair. Instead, the target device's architecture only allows interactions between certain qubits. This restricted connectivity is expressed through the device's *coupling graph* $A = (Q, E)$, where Q are *physical qubits* and $(Q_i, Q_j) \in E$ if a two-qubit gate can be executed between Q_i and Q_j. Coupling graphs can be *directed* or *undirected*. In the former case, executing a gate like the two-qubit CNOT on a directed edge (Q_i, Q_j) can only be performed with Q_i as the first and Q_j as the second qubit. In the following, we only consider undirected coupling graphs as our ideas can be straightforwardly extended to the directed case. For the sake of brevity we will use the term *architecture* when referring to the coupling graph of a quantum computing architecture.

To run a quantum algorithm on a target device, the logical qubits q first have to be mapped to the physical qubits Q. A *qubit assignment* is an injective function $\alpha : q \to Q$, i.e., a function that uniquely assigns each logical qubit a physical qubit. As long as $|q| \leq |Q|$, such a mapping can always be obtained. Executing a quantum circuit $G = g_0 \cdots g_{|G|-1}$ on a device with coupling graph A and initial assignment α is only possible if all two-qubit gates of the circuit act on qubits connected on the architecture. If this is not possible, the assignment has to be changed dynamically throughout the circuit in order for each gate of G to be executable. This is generally accomplished by adding SWAP gates to the circuit, which exchange two qubits in an assignment.

The mapping problem for a quantum circuit is to find an initial assignment α and a (potentially empty) sequence of SWAP insertions such that the entire circuit can be executed on the target architecture. We denote the optimal, i.e. minimal, number of swaps when mapping a quantum circuit G to architecture A with $s_{\text{opt}}(G, A)$.

Example 1. *Assume the circuit shown in Fig. 1a is to be mapped to the architecture defined by the coupling graph shown in Fig. 1b. No initial assignment exists such that this circuit can be executed without inserting* SWAP *gates. Taking the initial assignment α depicted on the left-hand side of Fig. 1c allows the direct execution of the first three* CNOT *gates. However, the* CNOT *between logical qubits q_3 and q_0 cannot be executed with this assignment. Inserting a* SWAP *between physical qubits Q_1 and Q_0 followed by a* SWAP *between physical qubit Q_1 and Q_2 leads to the assignment*

$$\{q_3 \mapsto Q_3, q_2 \mapsto Q_1, q_1 \mapsto Q_0, q_0 \mapsto Q_2\}.$$

Therefore, the CNOT *between q_3 and q_0 translates to a CNOT between Q_3 and Q_2 which can be directly executed.*

It is easy to see, that mapping a circuit can always be done by choosing some initial layout and greedily inserting SWAP gates. This leads to circuits with many SWAP gates—an undesirable property on NISQ devices due to the relatively high error rate of two-qubit gates. However, finding good solutions to the mapping problem is a challenging task in general and has even been shown to be NP-hard [7], [15], [19], [20]. Although these works make slightly different assumptions about the qubit mapping problem, the problem's complexity stays the same overall.

3 CONSIDERED PROBLEM

Given an n-qubit quantum circuit G and an architecture A (with $|A| \geq n$), the search space of the mapping problem is spanned by:

(1) *Qubit allocation:* Which subarchitecture A' of A consisting of at least n qubits shall be considered for the subsequent routing?
(2) *Qubit routing:* Assuming that $|A'|$ qubits have been allocated, which of the $|A'|!$ permutations—realizing arbitrary transitions between assignments via sequences of SWAP gates—to consider in front of every two-qubit gate?

As evident from the above, the search space in quantum circuit mapping grows exponentially with respect to the number of physical qubits allocated for the mapping process, i.e., the size of the considered subarchitecture. Thus, pruning parts of this search space is absolutely crucial for aiding mapping methods in efficiently traversing the search space. However, while limiting the search space, in general, increases the efficiency of mapping techniques (since a smaller search space is easier to explore), it bears the risk of cutting off regions which contain optimal/efficient solutions. Consequently, extra care has to be taken when

(1) reducing the number of allocated physical qubits $|A'|$, or
(2) limiting the considered permutations during qubit routing.

Most existing techniques that try to efficiently find good mapping solutions tend to focus on the second aspect, i.e., limiting the number of considered permutations or combinations of SWAPs during qubit routing [9], [13], [21]–[27]. On the other hand, the first aspect has hardly been considered thus far.

Example 2. *In order to get an impression of the qubit allocation problem, Fig. 2 shows all non-isomorphic subgraphs of the* 16*-qubit* ibmq_guadalupe *architecture—grouped by their number of qubits. Assume that a* 9*-qubit circuit shall be mapped to this architecture. Then, every subgraph with at least* 9 *qubits is a potential candidate for mapping—amounting to a total of* 91 *options.*

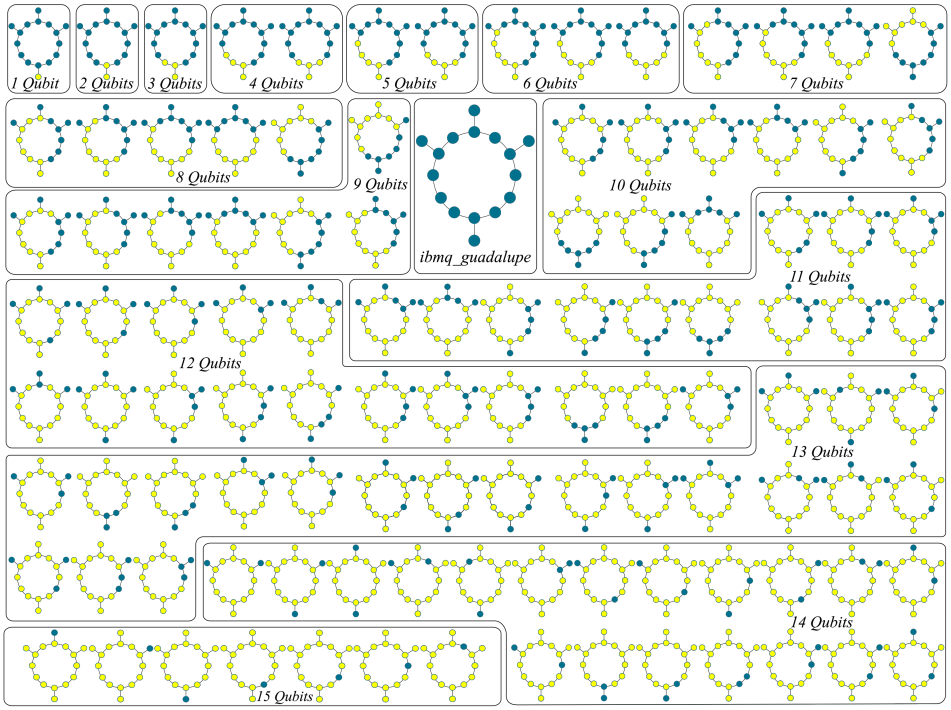


Fig. 2. Non-isomorphic subgraphs (marked as ◯) of the 16-qubit *ibmq_guadalupe* architecture (shown in the middle) grouped by their number of qubits.

Ideally, one would strive to allocate as few qubits as possible, in particular only as many as there are logical qubits in the circuit. This would entail the greatest search-space reduction as only as many qubits are allocated as are strictly necessary to ensure the existence of a mapping. Cutting off parts of the search space might, however, have the unwanted effect of eliminating optimal or even efficient solutions. Experimental evaluations on optimal quantum circuit mapping suggested that this is not the case [13], [14].

This work remedies this misconception by showing that considering larger than strictly necessary subarchitectures can be beneficial when searching for efficient solutions. This shows, once and for all, that considering subarchitectures of minimal size is not sufficient. However, instead of being content with a negative result, we then explore the following problem.

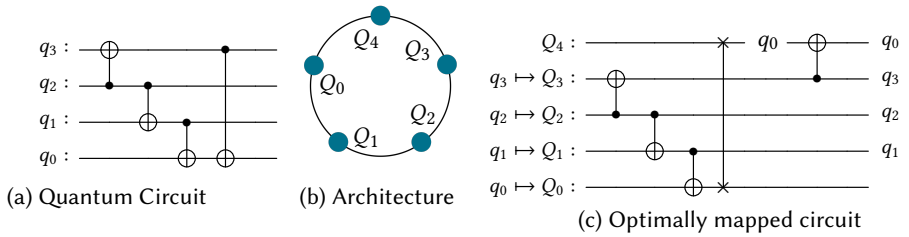
Central Problem. *Let A' and A'' be two subarchitectures of an architecture A. A' is said to possess* higher coverage *than A'', i.e., $A'' <^n_{cov} A'$ with respect to $n \in \mathbb{N}$ if:*

- *Any quantum circuit G over n qubits can be optimally mapped to A' with no more* SWAP*s than are needed for mapping G to A'', i.e., $\forall G : s_{opt}(G, A') \leq s_{opt}(G, A'')$.*
- *There is a quantum circuit G over n qubits that can be mapped to A' using less* SWAP*s than required for mapping it to A'', i.e, $\exists G : s_{opt}(G, A') < s_{opt}(G, A'')$.*

If only the first property is fulfilled we write $A'' \leq^n_{cov} A'$.

The problem of finding the optimal subarchitectures of A for a given size n is the problem of finding the smallest (with respect to the size of the subarchitectures) maximal elements with respect to $<^n_{cov}$, i.e.,

$$\min_{|A'|} \max_{\leq^n_{cov}} \{A' : A' \sqsubseteq A\}.$$



(a) Quantum Circuit (b) Architecture (c) Optimally mapped circuit

Fig. 3. Optimal solution to the mapping problem from Fig. 1 considering the complete architecture

4 EXISTENCE OF BETTER MAPPINGS ON LARGER SUBARCHITECTURES

In this section, we show that considering larger subgraphs can, under certain circumstances, lead to better solutions for the mapping problem. To this end, Section 4.1 and Section 4.2 establish two criteria that allow for ranking subarchitectures according to their coverage. These criteria are constructive in the sense that they allow for the definition of a method for computing good subarchitectures (with respect to $<^n_{\text{cov}}$)—trading off subarchitecture size and, by extension, search space size for the mapping problem, and likelihood of eliminating efficient mapping solutions. Afterwards, we give a complete characterization of the smallest maximal elements with respect to $<^n_{\text{cov}}$ among all the non-isomorphic subarchitectures of a given architecture. This proves that merely considering the subgraphs A' of an architecture A with as many qubits as the circuit to be mapped can lead to situations where the optimal solution to the mapping problem is no longer contained in the search space induced by the allocated qubits.

The proofs in this work require somewhat technical circuit constructions. Instead of listing the details here, an intuition of these constructions will be given. These can be generalized to arbitrary (sub-)architectures.

4.1 Subarchitectures with Shorter Connections

The first criterion is based on the observation that subarchitectures with shorter connections between certain qubits potentially allow for better mappings—even if they involve more qubits. An example illustrates the idea:

Example 3. *Consider again the circuit shown in Fig. 1a and assume it shall be mapped to the 5-qubit ring architecture shown in Fig. 1b. Previously, in Example 1, the circuit has been mapped to the 4-qubit line $Q_0 - Q_1 - Q_2 - Q_3$, which resulted in the addition of two SWAP gates in order to satisfy the coupling constraints (as illustrated in Fig. 1c). It can be shown that this is the optimal number of SWAP operations given that only four of the architecture's qubits are considered.*

If, however, the whole architecture is considered (i.e., including the idle qubit Q_4), the optimal solution just requires a single SWAP. The resulting circuit is shown in Fig. 3.

Based on the previous example, it is clear that extending a subarchitecture to a larger one might admit a better mapping for a quantum circuit if the path between some qubits is shorter in the larger subgraph. This insight suggests a natural way of ordering the subarchitectures of an architecture:

Definition 1. *Let the (partial) order $\preceq$ be defined as a relation on graphs with the following properties:*
Let $A = (V_A, E_A)$ and $A' = (V_{A'}, E_{A'})$ be two graphs. Then, $A' \prec A$ iff $A' \sqsubseteq A$ with subgraph isomorphism h and there are two nodes that have a shorter path between them in A than the corresponding nodes in A', i.e.,

$$\exists\, v, w \in V_{A'} : |p(v, w)| > |p(h(v), h(w))|.$$

If A' and A are only subgraph isomorphic with no shorter paths then we write $A' \preceq A$.

This order has the desirable property that $A' \preceq A$ iff the optimal mapping of an arbitrary quantum circuit G to the larger architecture A is at least as good as the mapping of G to subarchitecture A', i.e., $A' \preceq_{\text{cov}}^{|A'|} A$. The reason for this is simple: Since A' is a subarchitecture of A, any quantum circuit G mapped to A' can be mapped to A in exactly the same fashion.

Moreover, this partial ordering provides a way of ranking the subarchitectures of an architecture:

Theorem 4.1. *Let A' be a subarchitecture of an architecture A such that $A' \prec A$. Then, a quantum circuit G exists that is cheaper to map to A instead of A', i.e., $s_{opt}(G, A) < s_{opt}(G, A')$ and, by extension, $A' <_{cov}^{|A'|} A$.*

Proof Sketch. The intuition behind the proof is that, given $A' \prec A$, there exist vertices v and w in A' that have a shorter connection in A. If, during mapping, an interaction between v and w is required, this interaction can be realized more efficiently in A due to the shorter connection. □

Example 4. *Consider again the scenario from Example 3. Then, the 4-qubit line is strictly smaller than the 5-qubit ring with respect to $\preceq$ (i.e., `line` $\prec$ `ring`) since it holds that*

$$|p_{line}(Q_0, Q_3)| = 3 > 2 = |p_{ring}(Q_0, Q_3)|.$$

As a consequence of Theorem 4.1, there exists a circuit (e.g., the one shown in Fig. 1a) which has a more efficient mapping solution on the 5-qubit ring (see Fig. 3) as compared to the 4-qubit line (see Fig. 1c).

For any graph A and subgraph A', the set of *desirable* subarchitectures $\mathcal{D}(A', A)$ is given by the maximal elements with respect to $\prec$, i.e.,

$$\mathcal{D}(A', A) = \left\{D \colon\ A' \sqsubseteq D \sqsubseteq A \text{ and } \nexists D' \sqsubseteq A \colon\ A' \prec D' \wedge D \prec D'\right\}.$$

This set contains all subarchitectures worth considering during mapping due to their potential of providing a better mapping solution according to the ranking defined by the order $\preceq$.

4.2 Subarchitectures without Shorter Connections

As shown above, a better mapping can potentially be achieved if two qubits have a shorter connection on a larger subarchitecture than the one originally considered. The natural follow-up question is: Can a single additional qubit, that does *not* lead to a shorter connection between some qubits, bring any improvements in mapping a particular quantum circuit? In the following, we give an affirmative answer. The main idea is that, given an architecture A and a subarchitecture A', a larger subarchitecture $A' \sqsubseteq A'' \sqsubseteq A$ can allow for a better mapping, if it contains at least one more subarchitecture of size $|A'|$ that is not isomorphic to A'. Again, an example illustrates the idea:

Example 5. *Instead of a 5-qubit ring (as in Example 3), consider an architecture as shown on the bottom of Fig. 4. This architecture contains two non-isomorphic 4-qubit subarchitectures as shown in the middle of Fig. 4—a "line" and a "T"-shaped subarchitecture. The quantum circuits G and G' shown on top of the subarchitectures can be mapped to the respective subarchitecture without SWAPs, while they require a single SWAP to be realized on the other subarchitecture.*

Now, consider the quantum circuit $G^4G'^4$, i.e., a circuit composed of four repetitions of each of the two circuits described above. Mapping this circuit to either of the 4-qubit subarchitecture requires at least four SWAPs (since every repetition of the circuit which was not designed for the respective subarchitecture introduces a SWAP). However, it only takes three SWAPs to transform the qubit assignment on the "line" subarchitecture to a qubit assignment on the "T"-shaped subarchitecture on the whole architecture. Therefore, considering the whole architecture allows for a better mapping than mapping to either subarchitecture.

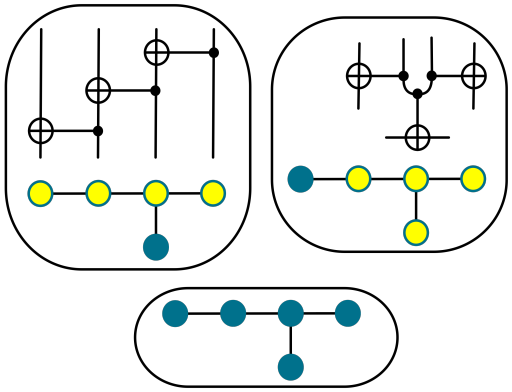


Fig. 4. 5-qubit architecture (bottom), its 4-qubit subgraph candidates (marked as), and two circuits each of which can be mapped to one of the subgraphs without SWAPs (top).

The following theorem formalizes this observation:

Theorem 4.2. *Let A', A'' be equally-sized (i.e., $|A'| = |A''| = n$), non-isomorphic (i.e., $A' \ncong A''$) proper induced subarchitectures of an architecture A (i.e., $A' \sqsubset A$ and $A'' \sqsubset A$).Then there is a quantum circuit G that is cheaper to map to A than to each individual subarchitecture, i.e., $A' <^n_{cov} A$ and $A'' <^n_{cov} A$.*

Proof Sketch. The intuition behind this proof is that, similar to Example 5, for each subarchitecture, a circuit can be constructed, that does not require any SWAPs to be executed on that subarchitecture, but will require at least one SWAP on the other subarchitecture. In addition, there is a certain number of SWAPs S that is needed to transform an assignment to A' to an assignment to A'' on the whole architecture. If G' (G'') is the circuit corresponding to A' (A''), then the circuit $G = (G')^{S+1}(G'')^{S+1}$ is cheaper to map to A than to either A' or A''. □

Thus, even a single additional qubit that does not induce a shorter connection may yield a better overall mapping.

4.3 Characterizing Optimal Subarchitectures

In the above, the benefit of an additional qubit is due to different parts of a circuit being more efficiently executable on different subarchitectures. This implies that all the desirable subarchitectures need to be covered in order to not lose optimality. The following example demonstrates, that extra care needs to be taken when determining such a covering.

Example 6. *The architecture A shown in Fig. 5 has, among others, the three 5-qubit subarchitectures labeled A', A'' and A'''. It is immediate that A'' and A''' are isomorphic. According to Theorem 4.2, there exists a 5-qubit circuit G that can be more efficiently mapped to a subarchitecture of A containing A' and A'' than to each individual subarchitecture. There are two possible options for covering both subgraphs on A by either connecting $A' - A''$ or $A' - A'''$. However, since the connection between A' and A'' is shorter, it has to be ensured that this graph is chosen over the alternative.*

Thus, a truly-optimal subarchitecture not only needs to cover any of the desirable subarchitectures but also has to allow for the shortest possible transformation between them. The last piece of the puzzle to characterize optimal subarchitectures of an architecture is that merely considering subarchitectures still does not suffice to guarantee that the optimal solution is not lost.

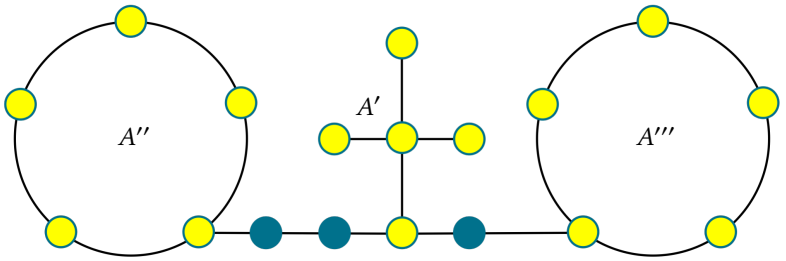


Fig. 5. Subarchitecture transformations

Example 7. *Consider again the circuit in Fig. 1a. Example 3 showed that this circuit can be more efficiently mapped to a* 5*-qubit ring than to a* 4*-qubit line. In Section 4.1, this was attributed to the shorter connection between two qubits in the larger architecture. A more general observation is that any arrangement of (logical) qubits on the device can potentially be permuted more efficiently on the larger architecture. Similarly, in Example 5, the* 5*-qubit architecture allows for the transformation of any linear arrangement of qubits to a "T"-shaped arrangement, which was not possible on either of it subgraphs.*

This brings us to the final characterization and theorem for optimal subarchitectures. Unfortunately, the precise formulation and proof of this theorem are very technical. For the sake of brevity, an informal version of the theorem is listed here.

Theorem 4.3. *The n-qubit optimal subarchitectures· of an architecture are those that allow to realize any sequence of n-qubit arrangements as efficiently as it could be realized on the whole architecture.*

This is an extremely restrictive requirement! There are way more arrangements of n qubits than there are n-qubit subarchitectures of an architecture. For example, while there is only one 4-qubit "line" subarchitecture up to isomorphism, there are twelve non-unique ways to arrange four qubits on a line. Any sequence of these arrangements needs to be realizable as efficiently on the potential subarchitecture than it would be on the whole architecture. This makes the "hunt" for optimal subarchitectures of an architecture look grim. It seems that not much is gained by characterizing optimal subarchitectures according to Theorem 4.3 as it still does not suggest a method for computing optimal subarchitectures. The real insight of this theorem is that it demonstrates just how general optimal subarchitectures are and that qubits cannot easily be neglected for the purpose of reducing the search space in quantum circuit mapping.

One might even be tempted to conclude that on any given architecture A one cannot neglect *any* physical qubits when mapping a quantum circuit G over $n < |A|$ qubits without potentially excluding essential parts of the search space. However, this pessimism is misplaced, as the following simple example shows.

Example 8. *Consider a* 5*-qubit "line" architecture. Then, the optimal mapping of any* 4*-qubit quantum circuit requires only four physical qubits. Theorem 4.3 explains why: None of the twelve possible arrangements of four qubits on a line can be transformed into each other more efficiently using the additional qubit.*

Computing optimal subarchitectures according to Theorem 4.3 is an incredibly complex problem due to the sheer number of possible arrangements. Fortunately, the requirements of Theorem 4.3 are way to strict in practice and it generally suffices to cover only the subarchitectures suggested by Theorem 4.1 and/or Theorem 4.2. How these near-optimal sets of subarchitectures are computed, is shown next.

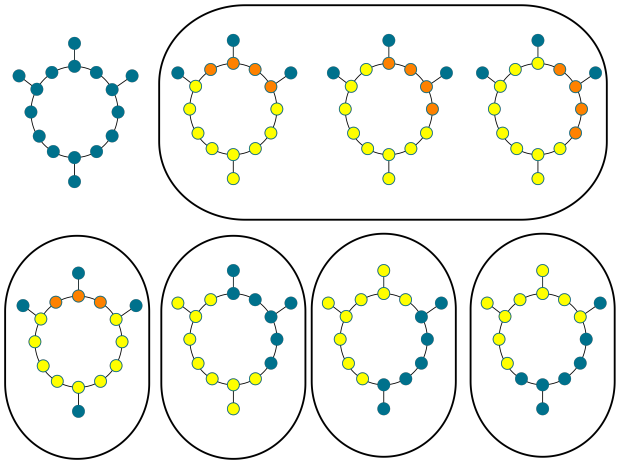


Fig. 6. *ibmq_guadalupe* architecture (top left), its non-isomorphic 9-qubit subarchitectures (marked as ●) and the corresponding desirable subarchitectures (with additional qubits marked as ● and qubits not included in the subarchitecture marked as ●)—forming the set of subarchitecture candidates for 9-qubit circuits. Multiple subarchitectures may lead to the same candidate.

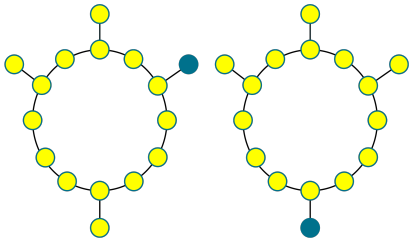


Fig. 7. Optimal subarchitecture candidates (marked as ●) for 9-qubit circuits on the *ibmq_guadalupe* architecture.

5 COMPUTING NEAR-OPTIMAL SUBARCHITECTURES

Based on the criteria derived in the previous section, we now demonstrate two ways of computing sets of subarchitectures to be considered for mapping which only depend on the number of qubits of the circuit to be mapped. As a result, for any particular architecture, these *subarchitecture candidates* can be computed a-priori without any particular information about the circuits.

5.1 Candidates for Optimal Subarchitectures

Assume that an n-qubit circuit G is to be mapped to an architecture A with $|A| \geq n$. Let $Sub_n(A)$ denote the set of all induced subarchitectures A' of A with size n, i.e.,

$$Sub_n(A) = \{A' : \ A' \sqsubseteq A \text{ and } |A'| = n\}.$$

According to Theorem 4.1, this set of subarchitectures is probably not a set of optimal subarchitectures since the architecture might contain "better" subarchitectures. Instead, for each of its elements, the corresponding set of *desirable* subarchitectures should be considered. Thus, an initial set of subarchitecture candidates $\text{Cand}_n(A)$ is given by

$$\text{Cand}_n(A) = \bigcup_{A' \in Sub_n(A)} \mathcal{D}(A', A).$$

Example 9. *Consider the* 16*-qubit* ibmq_guadalupe *architecture shown on the top left of Fig. 6 and assume that a* 9*-qubit circuit shall be mapped to this architecture. Then, Fig. 6 shows the architecture's seven non-isomorphic* 9*-qubit subarchitectures (marked as ●). On this architecture, the only way to decrease the distance between two qubits is to close the central ring of the architecture. In four out of seven cases, closing the ring indeed shortens the path between two qubits (marked as ●). Overall, the set of mapping candidates for* 9*-qubit circuits* $\text{Cand}_9(ibmq\_guadalupe)$ *consists of five subarchitectures ranging from nine to thirteen qubits.*

According to Theorem 4.2, if a subarchitecture subsumes two non-isomorphic smaller subarchitectures, then this larger subarchitecture needs to be considered in order to not lose out on optimality. Thus, the set of *optimal* subarchitecture candidates $OptCand_n(A)$ is given by the smallest subarchitectures of A (with respect to $\sqsubseteq$) that contain *all* subarchitecture candidates for n qubits, i.e.,

$$OptCand_n(A) = \min_{\sqsubseteq} \left(\underbrace{\bigcap_{C \in \mathrm{Cand}_n(A)} \overbrace{\{A' : \ C \sqsubseteq A' \sqsubseteq A\}}^{\text{subarchitectures of } A \text{ containing } C}}_{\text{intersection over all subarchitecture candidates}} \right)$$

This set is never empty, since, in the worst case, it consists of A itself (the entire architecture, per definition, has all subarchitecture candidates as subarchitectures). Although, in many cases, the elements of $OptCand_n(A)$ contain less qubits. Note that these candidates are optimal in the sense that no larger subarchitectures can be constructed using Theorem 4.1 and Theorem 4.2 that are an improvement with respect to $\leq^n_{\mathrm{cov}}$.

Example 10. *Consider again the ibmq_guadalupe architecture from the previous example. Then, Fig. 7 shows the optimal subarchitecture candidates for* 9*-qubit circuits. It is easy to check, that both indeed contain all five subarchitecture candidates illustrated in Fig. 6 and that removing any of their qubits would exclude one of them. While, in this case, only a single qubit is saved compared to the whole architecture, this still constitutes a significant reduction in the search space (e.g., by a factor of* 16 *due to only having to consider* 15! *instead of* 16! *possible permutations).*

By utilizing the derived subarchitectures to map quantum circuits, the search space can be greatly reduced without cutting away essential parts.

5.2 Adaptively Covering Subarchitecture Candidates

Although $OptCand_n(A)$ describes exactly those subarchitectures of an architecture A that are optimal with respect to Theorem 4.1 and Theorem 4.2, the requirement that these subarchitectures must contain *all* subarchitecture candidates is a very strong one. Since the resulting subarchitectures need to encompass many non-isomorphic subarchitectures, they frequently are large in size compared to the circuit to be mapped. Theorem 4.2 implies that this requirement cannot be dropped without potentially cutting away parts of the search space potentially containing optimal solutions. However, the structure of the circuits in the proof of Theorem 4.2 is still rather "artificial" since converting between different n-qubit subarchitectures in a single circuit is hardly relevant in practice. Thus, this criterion might be deliberately dropped in order to reduce the size of the subarchitectures to be considered—leaving only the subarchitecture candidates for mapping.

The number of subarchitecture candidates $\mathrm{Cand}_n(A)$ is, in general, bounded. In particular, if a architecture A with n qubits has diameter d, at most $\frac{n(n-1)}{2}(d-1)$ qubits can be added to improve the connectivity of the circuit. As architectures for currently existing devices exhibit low connectivity, the number of qubits that can be added to improve the connectivity of an architecture tends to be quite small. However, the sheer number of candidates can grow rather large and trying out every possible one can become exceedingly expensive—even if taking parallel execution into account. Thus, it might be desirable to find a collection of subarchitectures (with a limited number of elements) that covers *all* the subarchitecture candidates.

Definition 2. *Let A be an architecture and let* $\mathrm{Cand}_n(A)$ *denote the set of subarchitecture candidates for n-qubit circuits. Then, the set of covering candidates $CovCand_n(A)$ is given by the subarchitectures of A containing any of the subarchitecture candidates, i.e.,*

$$CovCand_n(A) = \bigcup_{C \in \mathrm{Cand}_n(A)} \{A' \colon C \sqsubseteq A' \sqsubseteq A\}$$

Moreover, a subarchitecture covering $Cov_n(A)$ *is a subset of $CovCand_n(A)$ with the property that for every subarchitecture candidate, there is at least one element in the covering that contains that candidate as a subarchitecture, i.e., $Cov_n(A) \subseteq CovCand_n(A)$ such that*

$$\forall C \in \mathrm{Cand}_n(A)\ \exists A' \in Cov_n(A) \colon C \sqsubseteq A'.$$

Algorithm 1 Compute subarchitecture covering

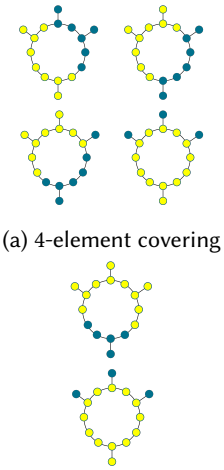
Input

A	Architecture
n	Number of qubits
k	Max. size of covering

Output

$Cov_n(A)$	Covering of A such that $\lvert Cov_n(A)\rvert \leq k$

Initialize $\mathrm{Cov}_n(A) \leftarrow \mathrm{Cand}_n(A)$
Initialize queue $\leftarrow \bigcup_{C \in \mathrm{Cand}_n(A)} \{A' \colon C \sqsubseteq A' \sqsubseteq A\}$
Sort queue with respect to the number of vertices.
while $\lvert \mathrm{Cov}_n(A)\rvert > k$ **do**
  $D \leftarrow \mathrm{pop}(\mathrm{queue})$
  Let $\mathrm{Cov}_D \leftarrow \{C \in \mathrm{Cov}_n(A) \colon C \sqsubseteq D\}$
  if $\lvert \mathrm{Cov}_D \rvert > 1$ **then**
    Update $\mathrm{Cov}_n(A) \leftarrow (\mathrm{Cov}_n(A) \setminus \mathrm{Cov}_D) \cup \{D\}$
  end if
end while



(a) 4-element covering

(b) 2-element covering

Fig. 8. Different subarchitecture coverings (marked as ◯) for 9-qubit circuits on the *ibmq_guadalupe* architecture.

Finding a minimal subarchitecture covering (with respect to the number of elements) is actually very easy, since the entire architecture, per definition, covers all subarchitecture candidates—at the disadvantage of having the largest number of qubits. In contrast, the set of subarchitecture candidates trivially forms the maximal covering—having the least number of qubits but the highest number of candidates. In order to determine a subarchitecture covering that is as small as possible while still containing subarchitectures over as few qubits as possible, we propose the greedy algorithm as sketched in Algorithm 1.

The algorithm starts off with the subarchitecture candidates as a covering and then iteratively picks a covering candidate (from smallest to largest). If the candidate covers more than a single architecture in the currently considered covering, the candidate replaces all covered architectures. This process is repeated until the size of the covering has been reduced to the desired size. This algorithm is guaranteed to terminate since, in the worst case, the entire architecture is returned.

The bottleneck of this algorithm is the computation of the non-isomorphic subarchitectures, as it requires solving many instances of the (sub-)graph isomorphism problem. During the computation

of the non-isomorphic subarchitectures, the set $\mathrm{Cand}_n(A)$ can be constructed on the fly. Because the ordering $\prec$ is dependent on the sub-graph isomorphism relation, it can be deduced during the sub-graph isomorphism check of two subarchitectures. If it is determined that $A' \sqsubseteq A''$, then it is simply a matter of determining whether there are two nodes in A'' that are closer in A'' than in A' to determine whether $A'' \prec A'$. The set $\mathcal{D}(A', A)$ can then be computed as the transitive closure.

Example 11. *Once more, consider the* ibmq_guadalupe *architecture from the previous examples. Then, Fig. 8 shows a four- and a two-element subarchitecture covering for* 9*-qubit circuits as determined by Algorithm 1. While the four-element covering shown in Fig. 8a contains smaller architectures (specifically, three* 9*-qubit, and one* 13*-qubit architecture), Fig. 8b shows that all three* 9*-qubit architectures can be covered by a single* 11*-qubit subarchitecture.*

Overall, Algorithm 1 allows one to adaptively cover all the subarchitecture candidates—offering a trade-off between the number of subarchitectures to consider and their respective size. The computed covering ensures that all practically relevant parts of the search space (i.e., excluding the pathological cases covered by Theorem 4.2 and Theorem 4.3) are considered during the subsequent mapping.

6 RESULTING TOOL

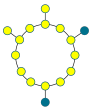
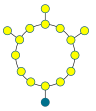
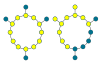
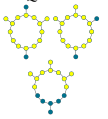
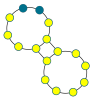
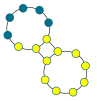
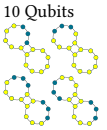
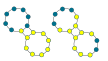
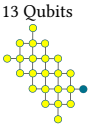
All of the above findings have been used to develop a Python-based tool that computes subarchitectures with a high coverage for a given architecture and size. In particular, this tool computes optimal subarchitecture candidates as discussed in Section 5.1 and minimal subarchitecture coverings using Algorithm 1. The tool is integrated into the quantum circuit mapping tool QMAP (https://github.com/cda-tum/qmap), which is part of the *Munich Quantum Toolkit* (MQT) [28]. All graph computations are performed using the retworkx library [29] which uses the VF2 algorithm [30] to solve the (sub-)graph isomorphism problem. Since the subgraph isomorphism problem is NP-hard [31], Algorithm 1 is inherently an exponential algorithm (at least as long as the question of P and NP is not settled). To compute all non-isomorphic subarchitectures of an architecture, one needs to first compute all subarchitectures of which there are an exponential number with respect to the size of the whole architecture. To demonstrate the runtime of computing optimal subarchitectures and the effect of mapping to subarchitectures, experimental evaluations were conducted on a 3.6 GHz Intel Xeon W-1370P machine running Ubuntu 20.04 with 128 GiB of main memory. Each evaluation's time limit was set to 24 hours. The current version of the tool performs all computations using a single thread. In principle, all non-isomorphic subarchitectures can be computed in parallel, which could accelerate the computation of optimal subarchitectures for larger architectures. In the sections that follow, we will show how the tool can be used to analyze state-of-the-art quantum computing architectures.

6.1 Optimal Subarchitectures and Coverings

The resulting tool was used to compute optimal subarchitecture candidates and minimal subarchitecture coverings for three representative quantum computing architectures: the 16-qubit *ibmq_guadalupe* architecture, another 16-qubit architecture consisting of two connected 8-qubit rings that serve as the foundation of Rigetti's quantum computing architectures, and a 23-qubit part of Google's Sycamore chip. More precisely, Table 1 lists the total number of connected subarchitectures and non-isomorphic subarchitectures for the three architectures, as well as the optimal subarchitecture candidates and minimal subarchitecture coverings for two different numbers of qubits for each device.

First, results confirm the findings illustrated before concerning the ibmq_guadalupe architecture. Since it is based on a 12-qubit ring, the optimal candidates from 8 qubits onward are all at least

Table 1. Example subarchitectures

Architecture	$\lvert A\rvert$	Connected Subarchitectures	Non-isomorphic Subarchitectures	Optimal Candidates		Coverings	
ibmq_guadalupe 	16	746	110	8 Qubits 	10 Qubits 	8 Qubits 	10 Qubits 
Rigetti 	16	1312	184	10 Qubits 	7 Qubits 	10 Qubits 	7 Qubits 
Sycamore 	23	300015	24786	7 Qubits 	13 Qubits 	7 Qubits 	13 Qubits  ⋮ + 95

of size 12 because that is the point at which there are shorter connections between qubits on the ring. Therefore it is quite hard to reduce the number of qubits of the considered subarchitectures during mapping on the *ibmq_guadalupe*. However, because the search space in quantum circuit mapping is exponential in the number of allocated qubits, every qubit saved by using Algorithm 1 significantly reduces the complexity of the mapping problem.

Things get more interesting with the 16-qubit Rigetti architecture which has a nice symmetry to it. The two central elements of this architecture are the 4-qubit ring in the middle and the 8-qubit rings attached to it. This is reflected in the covering graphs shown in Table 1, which can be roughly separated into subgraphs containing the 4-qubit ring and subgraphs containing the 8-qubit ring.

Finally, the Sycamore architecture by Google is an extreme case as it has a higher connectivity than any other superconducting quantum circuit architecture considered here. This is reflected in the large number of non-isomorphic subarchitectures contained in the 23-qubit architecture. Because of this high connectivity, one would expect that, according to Theorem 4.1, subarchitectures can be improved by adding additional qubits to complete the many 4-qubit rings present in the Sycamore architecture. And, indeed, it can be observed that, even for 13 qubits, the optimal subarchitecture candidate uses all but one of the architecture's qubits. Furthermore, it is hard to compute a small set of covering architectures. This makes the trade-off between coverage of optimal mapping solutions, complexity, and number of subarchitectures that need to be considered during quantum circuit mapping tricky. Even for the 13 qubit case, no covering with less than 97 elements can be found that does not contain a subarchitecture with more than 18 qubits. However, this is still a significant reduction from the 1153 desirable subarchitectures for 13 qubits.

Overall, Table 1 demonstrates just how hard it can be to reduce the size of subarchitectures considered for qubit allocation. This fact makes the theoretical findings provided in this work and the resulting tool especially appealing because they aid in understanding and tackling this difficult problem and provide a solid foundation for handling the enormous search space for future quantum circuit mapping techniques.

Table 2. Construction of the partial order $\prec$ for architectures of different sizes

Architecture	$\lvert A\rvert$	Connected	Non-isomorphic	t [s]
ibmq-lima	5	17	6	0.002
ibm-lagos	7	37	9	0.003
ibmq-guadalupe	16	746	110	1.380
ibmq-toronto	27	43721	5197	2544.954
sycamore-1-ring	4	13	4	0.000
sycamore-2-ring	6	40	9	0.001
sycamore-3-ring	8	117	24	0.012
sycamore-4-ring	9	218	28	0.020
sycamore-5-ring	11	609	111	0.368
sycamore-6-ring	13	1852	276	2.406
sycamore-7-ring	14	3343	383	4.592
sycamore-8-ring	16	9118	2004	93.152
sycamore-9-ring	18	27435	4156	402.590
sycamore-10-ring	19	49420	5364	665.922
sycamore-full	23	300015	24786	14 240.843
rigetti-1-ring	8	57	8	0.193
rigetti-2-ring	16	1312	184	2.176
rigetti-3-ring	24	31033	6181	1703.185
rigetti-4-ring	32	-	-	*timeout*

6.2 Computing the Partial Order $\prec$

To showcase the runtime scaling of computing optimal subarchitectures, consider Table 2. It illustrates the runtime with respect to the size of a given architecture and its number of different subarchitectures (isomorphic as well as non-isomorphic) when constructing the initial ordering $\prec$ of the subarchitectures—the computationally most expensive part of Algorithm 1. The considered architectures are comprised of different IBM quantum computers as well as versions of the Rigetti and Google architectures considered in the previous section. The Rigetti and Google architectures are comprised of 8- and 4-qubit rings, respectively, which are connected on a regular grid. For these architectures the number of connected rings for which $\prec$ has been computed, were varied to show the runtime dependency.

Architectures with more than 30 qubits could not be evaluated within 24 hours using the current version of the tool. This is mainly due to the current implementation not taking advantage of all available hardware resources, e.g., via parallelization. While there is naturally still a limit to the size of the considered architectures, a more optimized implementation can help analyze larger and more intricate architectures. The runtimes also suggest an optimization when mapping smaller circuits to regularly repeating architectures: instead of computing $\prec$ on the entire architecture, one can pick a smaller subarchitecture that is likely to contain most of the relevant subarchitectures for the circuit in question. This way, one can hope to gain a significant speedup whithout losing out on too many subarchitectures, as the runtime increases more drastically with the size of the architecture than the number of non-isomorphic subarchitectures.

Table 3. Optimal mapping with different subarchitectures

(a) 4-qubit circuits on 6-qubit architecture

Subarchitecture	#SWAP	t [s]
Two-local-random		
ring	9	0.290
line	12	1.077
fork	8	0.054
cover	8	3.228
full	8	11.798
Portfolio VQE		
ring	9	0.251
line	12	0.554
fork	8	0.038
cover	8	0.294
full	8	12.621
QFT Entangled		
ring	3	0.106
line	5	0.758
fork	6	0.043
cover	5	0.292
full	3	1.602
Realamprandom		
ring	9	0.449
line	12	2.563
fork	8	0.050
cover	8	1.556
full	8	4.359

(b) 5-qubit circuits on 7-qubit architecture

Subarchitecture	#SWAP	t [s]
Graphstate		
ring	1	0.055
line	3	0.086
fork	2	0.034
cover	2	0.138
full	1	61.213
AE		
ring	6	2.424
line	6	1.495
fork	4	0.581
cover	4	5.259
full	4	227.778
QPE Exact		
ring	6	79.959
line	7	10.945
fork	5	0.599
cover	5	115.252
full	5	1189.601
QGAN		
ring	6	0.873
line	6	0.884
fork	4	0.267
cover	4	1.482
full	4	171.095

6.3 Impact on Quantum Circuit Mapping

In order to illustrate the potential impact of considering optimal subarchitectures for quantum circuit mapping, Table 3 shows the effects of subarchitectures on the runtime and number of SWAPs when mapping quantum circuits. To this end, the exact mapper available in QMAP (https://github.com/cda-tum/qmap) [14] was used for exact mapping, and the benchmarks were taken from the benchmark library MQTBench (version 0.2.2) [32].

As architectures, a the 6- and 7-qubit architectures in Fig. 9 were considered. Both are comprised of a central ring of qubits with one additional qubit connected to one of the qubit rings. This way, the number of different non-isomorphic subarchitectures is kept small while seeing the effect of having shorter connections, different subarchitectures, and covering subarchitectures. There are only 5 subarchitectures to consider: the central ring (*ring*), the 4- (5-) qubit line (*line*), the 3- (4-) qubit line with an additional qubit connected to one of the middle qubits (*fork*), the subarchitecture containing both *line* and *fork* (*cover*), and the entire architecture itself (*full*). The optimal subarchitecture for 4- and 5-qubit circuits on both architectures is the entire architecture itself. However, from Table 3, we can see that mapping to the entire architecture is often quite costly in terms of runtime compared to mapping to subarchitectures. A 2-subarchitecture covering can be obtained by taking the *ring* and *cover* subarchitectures together. This often yields much better runtimes without losing the optimal solution because it is guaranteed to be contained in these two subarchitectures by Theorem 4.3.

This table also shows the validity of Theorem 4.1, as the additional qubit completing the ring actually helps to improve the quality of the mapping. According to Theorem 4.2, there are circuits that are cheaper to map to *cover* than to the two smaller architectures *line* and *fork*. This theoretical property was not observed in the experiments as the considered real-world quantum circuits where way too shallow and structured to possibly benefit from being mapped to *cover*.

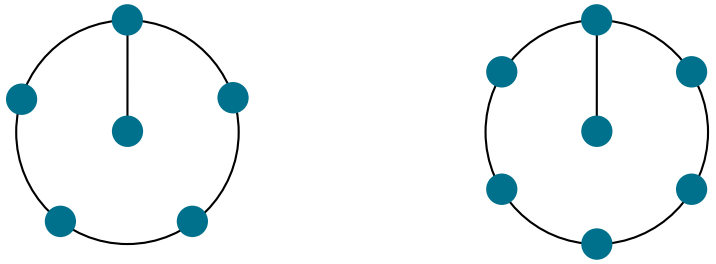


Fig. 9. Architectures for optimal quantum circuit mapping

7 CONCLUSION AND FUTURE WORK

In this work, we introduced the notion of optimal subarchitectures for mapping n-qubit quantum circuits and disproved a previous conjecture that all n-qubit quantum circuits can be mapped to some n-qubit subarchitecture of a quantum computing device without potentially eliminating optimal mapping solutions. In fact, quite the opposite is the case: trying to reduce the number of qubits considered in the qubit allocation process without cutting off essential parts of the search space in the subsequent mapping is incredibly difficult. Despite this theoretical result, the structure of the quantum circuits that require such large optimal subarchitectures is pretty artificial, and the conditions for optimality can be relaxed a bit. Hence, we introduced an algorithm for computing subarchitectures that constitutes a trade-off between coverage of optimality and architecture complexity. The resulting tool is integrated into the open-source tool QMAP (available at https://github.com/cda-tum/qmap), which is part of the *Munich Quantum Toolkit* (MQT).

Based on this first method for computing near-optimal subarchitectures, there are several possible directions for improvement:

- Quantum circuits used in real-world applications naturally possess a lot of structure. It might, therefore, be possible to compute optimal subarchitectures for certain *classes* of quantum circuits instead of any quantum circuit of a given size. These classes can probably be deduced from two-qubit interaction patterns repeatedly found in real-world quantum circuits.
- In this work, no distinction was made between isomorphic subarchitectures of a quantum computing device. In reality, neither the qubits nor the connections between them on an architecture are equally reliable. Consequently, placing a certain subarchitecture on a different part of the whole architecture might yield more reliable circuit executions. Tools like *mapomatic* [33] exist that search for low-noise subarchitectures given an already-compiled quantum circuit. Such information could additionally be included in the methodology for determining suitable subarchitectures to provide for quantum compilers that consider noise.
- In order to compute (near-)optimal subarchitectures of future large-scale quantum computers, more efficient algorithms than the one initially presented here will have to be developed. A promising approach is to take advantage of the highly symmetric structure of real-world quantum computing architectures.

To summarize, this work has laid the groundwork for further research into the problem of computing (near-)optimal subarchitectures of state-of-the-art quantum computers. This first work is a good starting point for more methods and improvements that can help develop quantum circuit mappers that can handle the mapping problem for large-scale quantum computers in an efficient way.

Acknowledgements

This work received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 101001318), was part of the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus, and has been supported by the BMWK on the basis of a decision by the German Bundestag through project QuaST, as well as by the BMK, BMDW, and the State of Upper Austria in the frame of the COMET program (managed by the FFG).

REFERENCES

[1] Michael A. Nielsen and Isaac L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[2] M. H. Devoret and R. J. Schoelkopf, "Superconducting circuits for quantum information: An outlook," *Science*, vol. 339, no. 6124, pp. 1169–1174, 2013.

[3] Mehdi Saeedi, Robert Wille, and Rolf Drechsler, "Synthesis of quantum circuits for linear nearest neighbor architectures," *Quantum Information Processing*, vol. 10, no. 3, pp. 355–377, 2011. DOI: 10.1007/s11128-010-0201-2. arXiv: 1110.6412.

[4] Alwin Zulehner, Alexandru Paler, and Robert Wille, "An efficient methodology for mapping quantum circuits to the IBM QX architectures," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2019.

[5] Alwin Zulehner and Robert Wille, "Compiling SU(4) quantum circuits to IBM QX architectures," in *Asia and South Pacific Design Automation Conf.*, 2019, pp. 185–190. DOI: 10.1145/3287624.3287704.

[6] Stefan Hillmich, Alwin Zulehner, and Robert Wille, "Exploiting Quantum Teleportation in Quantum Circuit Mapping," in *Design, Automation and Test in Europe*, 2021, pp. 792–797. DOI: 10.1145/3394885.3431604.

[7] Marcos Yukio Siraichi *et al.*, "Qubit allocation," in *Int'l Symp. on Code Generation and Optimization*, 2018, pp. 113–125. DOI: 10.1145/3168822.

[8] Pengcheng Zhu, Xueyun Cheng, and Zhijin Guan, "An exact qubit allocation approach for NISQ architectures," *Quantum Information Processing*, vol. 19, no. 11, p. 391, 2020.

[9] Alexander Cowtan *et al.*, "On the qubit routing problem," in *Theory of quantum computation, communication and cryptography*, 2019.

[10] Yunseong Nam *et al.*, "Automated optimization of large quantum circuits with continuous parameters," *npj Quantum Information*, 2018.

[11] Aleks Kissinger and John van de Wetering, "Reducing T-count with the ZX-calculus," *Physical Review A*, 2020.

[12] Prakash Murali *et al.*, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in *Int'l Conf. On Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 1015–1029. DOI: 10.1145/3297858.3304075.

[13] Lukas Burgholzer, Sarah Schneider, and Robert Wille, "Limiting the search space in optimal quantum circuit mapping," in *Asia and South Pacific Design Automation Conf.*, 2022.

[14] Robert Wille, Lukas Burgholzer, and Alwin Zulehner, "Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations," in *Design Automation Conf.*, 2019.

[15] A. Botea, A. Kishimoto, and Radu Marinescu, "On the complexity of quantum circuit compilation," in *Int'l Symp. on Combinatorial Search*, 2018.

[16] Jay Gambetta, "IBM's Roadmap For Scaling Quantum Technology," *IBM Research Blog*, 2020.

[17] John Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018.
[18] Adrian Bondy and Uppaluri Siva Ramachandra Murty, *Graph Theory*. Springer International Publishing.
[19] Dmitri Maslov, Sean M. Falconer, and Michele Mosca, "Quantum circuit placement: Optimizing qubit-to-qubit interactions through mapping quantum circuits into a physical experiment," in *Design Automation Conf.*, 2007, pp. 962–965. DOI: 10.1145/1278480.1278717.
[20] B. Tan and J. Cong, "Optimality study of existing quantum computing layout synthesis tools," *IEEE Trans. on Computers*, vol. 70, no. 09, pp. 1363–1373, 2021. DOI: 10.1109/TC.2020.3009140.
[21] Gushu Li, Yufei Ding, and Yuan Xie, "Tackling the qubit mapping problem for NISQ-era quantum devices," in *Int'l Conf. On Architectural Support for Programming Languages and Operating Systems*, 2019.
[22] Bochen Tan and Jason Cong, "Optimal layout synthesis for quantum computing," in *Int'l Conf. on CAD*, 2020.
[23] Animesh Sinha, Utkarsh Azad, and Harjinder Singh, "Qubit routing using graph neural network aided Monte Carlo tree search," 2022. arXiv: 2104.01992 [quant-ph].
[24] Alexandru Paler *et al.*, "Machine learning optimization of quantum circuit layouts," *ACM Transactions on Quantum Computing*, 2022. DOI: 10.1145/3565271.
[25] Marco Baioletti, Riccardo Rasconi, and Angelo Oddi, "A novel ant colony optimization strategy for the quantum circuit compilation problem," in *Evolutionary Computation in Combinatorial Optimization*, 2021.
[26] Matteo G. Pozzi *et al.*, *Using reinforcement learning to perform qubit routing in quantum compilers*, 2020. arXiv: 2007.15957 [quant-ph].
[27] Arighna Deb, Gerhard W. Dueck, and Robert Wille, "Exploring the potential benefits of alternative quantum computing architectures," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 40, no. 9, pp. 1825–1835, 2021. DOI: 10.1109/TCAD.2020.3032072.
[28] R. Wille and L. Burgholzer, "MQT QMAP: Efficient quantum circuit mapping," in *Int'l Symp. on Physical Design*, 2023.
[29] Matthew Treinish *et al.*, "Retworkx: A high-performance graph library for python," 2022. arXiv: 2110.15221 [cs].
[30] Luigi Cordella *et al.*, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1367–1372, 2004.
[31] Stephen A. Cook, "The complexity of theorem-proving procedures," in *Symposium on Theory of Computing*, 1971, pp. 151–158.
[32] Nils Quetschlich, Lukas Burgholzer, and Robert Wille, *MQT Bench: Benchmarking software and design automation tools for quantum computing*, 2022. arXiv: 2204.13719.
[33] Paul Nation and Matthew Treinish, *Mapomatic*. [Online]. Available: https://github.com/Qiskit-Partners/mapomatic.