Evaluating the Flexibility of A* for Mapping Quantum Circuits

Alwin Zulehner, Hartwig Bauer, and Robert Wille

Johannes Kepler University Linz, Austria
alwin.zulehner@jku.at, hartwig.bauer@jku.at, robert.wille@jku.at

Abstract. Mapping quantum circuits to real quantum architectures (while keeping the respectively considered cost as small as possible) has become an important research task since it is required to execute algorithms on real devices. Since the underlying problem is NP-complete, several heuristic approaches have been proposed. Recently, approaches utilizing A* search to map quantum circuits to, e.g., Nearest Neighbor architectures or IBM QX architectures have gained substantial interest. However, their performance usually has only been evaluated in a rather narrow context, i.e., for single architectures and objectives only. In this work, we evaluate the flexibility of A* in the context of mapping quantum circuits to physical devices. To this end, we review the underlying concepts and show its flexibility with respect to the considered architecture. Furthermore, we demonstrate how easy such solutions can be adjusted towards optimizing different design objectives or cost metrics by providing a generalized and parameterizable cost function for the A* search that can also be easily extended to support future cost metrics.

1 Introduction

Quantum computing [1] utilizes quantum mechanical effects like superposition and entanglement to allow for significant (in many cases exponential) speedups compared to current devices for applications like integer factorization [2], database search [3], or simulation of physical systems [4]. In the recent years, there has been a significant progress in the physical realizations of real quantum hardware. Arising from academic proof-of-concept realizations [5, 6], nowadays publicly available quantum computers are made accessible, e.g., by IBM through a cloud interface [7] and a first prototype for commercial use is available as well [8]. Moreover, architectures are envisioned to manage the step from current *Noisy Intermediate Scale Quantum* (NISQ [9]) devices to fault-tolerant ones composed of thousands of qubits [10, 11].

However, to run quantum algorithms on such real devices, the respective high-level operations have to be broken down into elementary operations (acting on one or two qubits only) supported by the hardware (e.g., using approaches such as [12–14]) and the logical qubits of the quantum algorithm have to be mapped to physical ones of the quantum device. Especially the mapping part constitutes a tough challenge since further physical constraints have to be considered. In fact, not all pairs of qubits may interact with each other due to so-called coupling-constraints. Hence, the mapping usually has to change dynamically throughout the execution of a quantum computation. This is achieved by adding so-called SWAP operations that exchange the state of two physical qubits and, by this, "move around" the logical qubits on the hardware. This overhead shall obviously be kept as small as possible since each additional operation increases the execution time and the possibility of an unreliable result (since

quantum computing is error prone yet)—resulting in a task that has recently been proven to be NP-complete [15, 16].

In the last decade, several solutions for this mapping problem have been proposed. First solutions focused on *Nearest Neighbor* (NN) architectures where the qubits are located in a 1- or 2-dimensional grid and only neighboring qubits may interact with each other [17–20]. With the appearance of publicly available quantum computers, researchers also started to focus on the mapping problem for IBM QX architectures—leading to further solutions dedicated for these architectures [21–26]. Many of the proposed approaches—for NN as well as for IBM QX architectures—have in common that they utilize the A* search algorithm. However, their performance usually has only been evaluated in a rather narrow context, i.e., for single architectures and objectives.

In this work, we investigate the flexibility of A*-based mapping and propose a generic approach that allows for an efficient mapping to NN as well as to IBM QX architectures while optimizing different design objectives. This is achieved by exploiting the fact that the constraints of different architectures can be modeled by coupling maps and by using a generic and parameterizable cost function. Given the coupling-constraints of any envisioned new architecture as well as appropriate parameters for the cost function, the proposed solution inherently provides a customized mapping algorithm without writing any code. Moreover, by slightly adjusting the cost function, future design objectives can be easily incorporated as well.

Our evaluations show that the resulting approach, although being generic and flexible with respective to different architectures and objectives, remains competitive even against state-of-the-art solutions which have been optimized over the last ten years and to a single architecture and a single objective. Moreover, the evaluations demonstrate that simply changing some few parameters (rather than developing new dedicated algorithms) allows to optimize for various design objectives like gate count, circuit depth, or an equally distributed workload for the qubits. Overall, this shows the flexibility of A*-based mapping of quantum circuits.

This paper is structured as follows. In Section 2, we review quantum circuits and quantum architectures including a description of the considered mapping problem. Section 3 discusses the A*-based mapping in general as well as its flexibility. Eventually, the proposed resulting generic approach is evaluated in Section 4, while Section 5 concludes the paper.

2 Quantum Circuits and Quantum Architectures

To keep the paper self-contained, this section briefly recapitulates quantum circuits as well as currently considered quantum architectures.

2.1 Quantum Circuits

In contrast to conventional computations, *quantum computations* [1] operate on qubits instead of bits. A *qubit* is a two-state quantum system, with basis states $|0\rangle \equiv \binom{1}{0}$ and $|1\rangle \equiv \binom{0}{1}$ (representing Boolean values 0 and 1, respectively). Furthermore, a qubit can be in a superposition of these basis states, i.e., $|x\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where the complex amplitudes α and β satisfy $|\alpha|^2 + |\beta|^2 = 1$. The state of a qubit can be modified by applying quantum operations, whose functionality can be described by 2×2-dimensional unitary matrices. Commonly used 1-qubit gates are

$$NOT = X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \text{ and } \quad T = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1+i}{2} \end{bmatrix},$$

which invert the state of a qubit, sets it into a superposition, or conducts a phase shift by $\frac{1+i}{\sqrt{2}}$, respectively.[1] The state of a qubit cannot be directly observed. Instead, measurement collapses the qubit into one of the two basis states $|0\rangle$ or $|1\rangle$. More precisely, the qubit collapses to basis state $|0\rangle$ with probability $|\alpha|^2$ and to basis state $|1\rangle$ with probability $|\beta|^2$.

The above extends to quantum systems composed of n qubits. Here, due to a quantum mechanical effect called *entanglement*, the state of a qubit might additionally be influenced by other qubits.[2] Hence, the qubits can not be considered individually, rather as complete system with 2^n basis states and corresponding amplitudes. The state of such a system is then accordingly manipulated by a $2^n \times 2^n$-dimensional unitary matrix. Since such operations acting on all qubits can not be realized physically, they are usually decomposed into a sequence of operations that act on one or two qubits only (other qubits are not affected). For example, it has been shown that—besides arbitrary single-qubit operations—having a controlled NOT (i.e., CNOT) operation

$$CNOT = CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

where the state of the *target qubit* is inverted if the *control qubit* is in its basis state $|1\rangle$, is sufficient to allow for universal quantum computing. Hence, any $2^n \times 2^n$ unitary matrix can be decomposed into a sequence composed of 1-qubit operations and CNOTs.

A commonly used representation for quantum computations are *quantum circuits*. Here, the respective qubits are denoted by horizontal *circuit lines*. Operations are represented by *quantum gates*. Boxes labeled with the respective functionality denote 1-qubit gates, whereas $\bullet$ and $\oplus$ denote the control and target qubit of a CNOT gate, respectively. Overall, this yields a representation of a quantum circuit as a cascade $G = g_1 g_2 \dots g_{|G|}$ of gates (drawn from left to right), where $|G|$ denotes the total number of gates. The number of qubits and, thus, the number of circuit lines is denoted by n.

Example 1 *Figure 1 shows a quantum circuit composed of $|G| = 22$ gates and $n = 5$ circuit lines. Each circuit line represents a qubit q_0 – q_4. The first (leftmost) gate describes a CNOT operation with control line q_1 and target line q_0. The U blocks represent single qubit operations.*[3]

Since there are various ways to realize certain quantum functionality by means of a quantum circuit, one has to define cost metrics that allow designers to chose the best realization. Commonly used cost metrics are:

- *Gate count $gc(G)$:* The gate count of a circuit G is its number of elementary gates. When using weights for each gate type, this also allows to estimate the fidelity of the overall circuit.
- *Circuit Depth $cd(G)$:* The depth of a circuit G describes the minimal number of time-steps required to execute all gates. In this work, we assume that all elementary operations require one time-step and that operations acting on

[1] The new state of the qubit is determined by multiplying the corresponding state vector and the unitary matrix [27].

[2] Albert Einstein referred to this effect as *spooky action at a distance*.

[3] Note that we do not further specify the functionality of the single qubit gates since it is irrelevant for the mapping process.

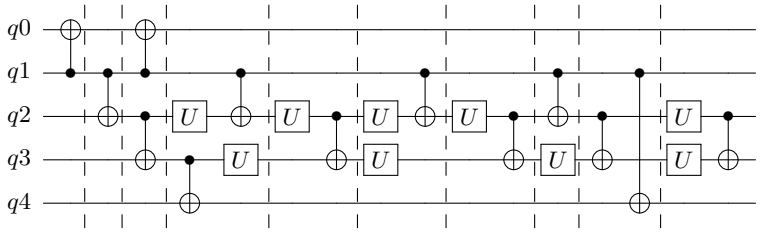

Fig. 1. Quantum circuit

disjoint sets of qubits can be executed in parallel. However, these assumptions can be easily adjusted if desired.

Besides these established cost metrics, we define another (artificial) cost metric to demonstrate the flexibility of A*-based mapping of quantum circuits, e.g., when extending the cost function to take qubit fidelity [28] into account in the future.

– *Workload Distribution* $wd(G)$: The workload $wd'(q_i)$ of a qubit q_i is determined by the number of gates that act on q_i (or use it as a control). Workload distribution of a circuit G is then defined as standard deviation of the workload for each qubit (that is affected by at least one gate), i.e.,

$$wd(G) = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(wd'(q_i) - wd_{avg})^2}.$$

Example 1 (continued) *As stated above, the gate count of the circuit G shown in Figure 1 is $gc(G) = 22$. The depth of this circuit $cd(G) = 15$. For example, the CNOT with control q_1 and target q_0 (i.e., $CNOT(q_1, q_0)$) can be applied simultaneously with the gate $CNOT(q_2, q_3)$. The gate $CNOT(q_3, q_4)$ has to be applied later because it operates also on q_3. Finally, the workload distribution is* $wd(G) = \sqrt{\frac{108}{5}} = 4.65$.

2.2 Mapping Quantum Circuits to Quantum Architectures

In the recent years, there has been a significant progress in the physical realization of real quantum hardware. Arising from academic proof-of-concept realizations [5, 6], there are already publicly available quantum computers made accessible by IBM through a cloud interface [7] as well as first commercially available ones [8]. Moreover, architectures are envisioned to manage the step from current *Noisy Intermediate Scale Quantum* (NISQ [9]) devices to fault-tolerant ones composed of thousands of qubits [10, 11]. However, all these architectures come with certain restrictions regarding (1) the available elementary quantum operations and (2) the allowed qubit connectivity (i.e., which pairs of qubits may interact with each other by means of two-qubit gates). These restrictions have to be considered when executing quantum circuits on them.

Since decomposition into different gate libraries is already well covered by literature (see, e.g., [12–14]), we focus on the connectivity constraints of the architectures in the following. Before real quantum computers became available,

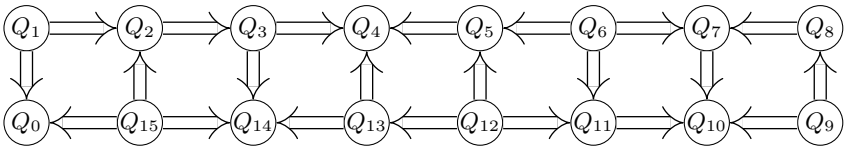

Fig. 2. Coupling map for IBM QX5 [30]

researchers considered so-called *Nearest Neighbor* (NN) architectures, where the qubits are arranged in a 1- or 2-dimensional grid and interactions are only possible between neighboring qubits.[4] However, IBM's QX architectures—the first quantum computers made publicly available—employ slightly different constraints. While interactions are also only possible between certain pairs of qubits, their layout is not necessarily as regular as a 1- or 2-dimensional grid and it is additionally given which qubit may act as control and which qubit may act as target (i.e., the *direction* of the CNOT is fixed). These constraints are defined by so-called *coupling maps* (i.e., a directed graph), where the *m physical qubits* $Q_0, Q_1, \ldots Q_{m-1}$ are represented by vertices and an arrow from Q_i to Q_j indicates that a CNOT with control qubit Q_i and target qubit Q_j can be executed (these constraints are denoted *coupling-constraints* in the following). In the following, we consider architectures specified by coupling maps, since this approach is more general (constraints of NN architectures can be easily modeled by an according coupling map as well).

Example 2 *Figure 2 shows the coupling map of IBM's QX5 [30] architecture. As described above, the arrow from Q_1 to Q_2 represents that CNOTs with control Q_1 and target Q_2 can be applied. It also means that a CNOT with control Q_2 and target Q_1 is not possible. Since there is also no arrow connecting Q_1 and Q_3, a CNOT cannot be applied on these two qubits (independent of its direction).*

In order to execute quantum circuits on architectures as described above, two steps are conducted. First, the quantum gates of the circuit are decomposed into elementary operations available on the hardware. Since this step is already well covered in the literature [12–14], we assume that is has already been conducted. The second step—mapping the n logical qubits $q_0, q_1, \ldots q_{n-1}$ of a quantum circuit to the m physical qubits $Q_0, Q_1, \ldots Q_{m-1}$ of a quantum computer while satisfying all *coupling-constraints*—constitutes a tougher challenge. Usually it is not possible to find a mapping that satisfies the constraints throughout the whole circuit. This becomes immediately clear by considering a circuit where one qubit interacts with more other qubits than the maximal degree of a coupling map. Assuming an initial mapping, the following problems may occur:

- A CNOT shall be applied where the control and the target qubit are mapped to physical qubits that are not connected in the coupling map.
- A CNOT shall be applied where the control and the target qubit are mapped to physical qubits that are connected in the coupling map, but there is only a connection in the "wrong" direction.

To overcome these issues, the mapping procedure has to change the mapping dynamically by inserting additional operations. The most established technique is to insert so-called SWAP operations that exchange the state of two physical qubits and, thus, move around the logical qubits—changing their mapping dynamically.[5]

[4] Note that this constraint is still valid for many recent architectures, e.g., Google's *Bristlecone* relies on such a 2D architecture [29].

[5] Note that there also exist other methods to overcome the problems [25], but they tend to generate larger overhead for bigger circuits.

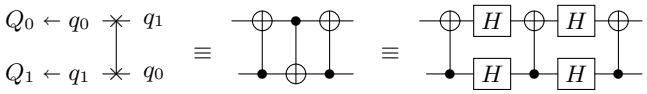

Fig. 3. SWAP operation

Example 3 *Figure 3 shows a SWAP operation that swaps the state of the physical qubits Q_0 and Q_1 of IBM QX5. Since the logical qubits q_0 and q_1 are mapped to Q_0 and Q_1 initially, the SWAP operations changes the mapping such that q_0 and q_1 are mapped to Q_1 and Q_0 afterwards. The SWAP operation is decomposed into three CNOTs as shown in the middle of Figure 3. Since only CNOTs with control Q_1 and target Q_0 are possible (cf. Figure 2), the direction of the middle CNOT has to be switched. This is achieved by inserting Hadamard gates before and after this CNOT.*

While SWAP operations are sufficient to overcome both issues listed above, the second one can be handled with fewer overhead. Like in the decomposition of a SWAP operation shown in Example 3, the direction of a CNOT can be switched by inserting four Hadamard operations. Minimizing the overhead (e.g., regarding one of the cost metrics defined in Section 2.1) caused by satisfying the coupling-constraints has recently been proven to be an NP-complete problem [15, 16].

Since the mapping problem is NP-complete, several heuristic approaches have been proposed. These include dedicated solutions for NN architectures [17–20] or for real ones [21–26] (e.g., IBM's QX architectures) that are specified by coupling maps and usually focus on optimizing the gate count of the mapped circuit. Since many of these algorithms are based on an A* search, we analyze and evaluate the flexibility of an A*-based mapping in this work.

3 Mapping Quantum Circuits Using A*

This section discusses the flexibility of A*-based search methods for mapping quantum circuits to quantum architectures. To this end, we first sketch the general idea and, afterwards, provide the details of the A*-based mapping algorithm. Based on that, we discuss how easily the approach can be extended for different architectures and objectives.

3.1 General Idea

This section briefly lines out the general idea of mapping quantum circuits to real architectures using the A* search algorithm. Since solving the problem in an exact fashion (i.e., with minimal overhead) has been proven to be NP-complete [15, 16], we aim for a heuristic approach to provide a solution within reasonable time.

The general idea is to partition the circuit to be mapped into k sub-circuits $G_0, G_1, \ldots G_{k-1}$. These sub-circuits are formed in a way, such that there exists a mapping from the logical qubits of the sub-circuit to the physical qubits of the target architecture where all coupling-constraints given by the coupling map are satisfied (neglecting the direction of the CNOTs since this is easily adjusted by inserting four Hadamard operations). Having that, no SWAP operations have to be inserted inside the sub-circuits (only H operations may be required). In between the sub-circuits, *permutation sub-circuits* composed of SWAP operations are inserted that change the mapping of logical qubits to physical ones dynamically. Determining the cheapest permutation circuit (with respect to a

given cost function) such that all coupling-constraints are satisfied for the next sub-circuit to be mapped (again, neglecting the direction of the connections between physical qubits) is conducted using an A* search. Hereby it is notable that the cost function might include a look-ahead for future sub-circuits such that the overall cost are subject to be optimized rather than utilizing locally-optimal permutations (which often leads to an increase of the overall cost [22]).

One flexibility of the proposed mapping algorithm is how to form the sub-circuits. In the literature, there exist approaches using different strategies. The most straightforward and naive version is to treat each gate as its own sub-circuit. Then, the A* search algorithm is called once for each gate (except for the first one). To reduce the number of calls to the search algorithm and to optimize the overall cost by explicitly considering multiple gates, sub-circuit composed of several gates are usually considered. One commonly used possibility is to group all gates that act on disjoint qubits into a sub-circuit [21, 22].[6] Alternatively, it is also possible to group as many gates into a sub-circuit such that a satisfying mapping can still be found for the sub-circuit (e.g., using SAT solvers as done in [20, 26]). Finally, it is also possible leave the decision of determining the sub-circuits open for the A* search as done in [23]. Here, a set of possible gates to be grouped are passed to the search algorithm, which inherently chooses a subset of these gates to be included in the next sub-circuit according to its objective function.[7]

Example 4 *Considering the circuit shown in Figure 1, the partitioning into sub-circuits based gates acting on disjoint qubits [21, 22] is conducted as follows when ignoring 1-qubit gates. The first sub-circuit G_0 contains the CNOT gate with control q_1 and target q_0, i.e., $CNOT(q_1, q_0)$. The second gate of the circuit $CNOT(q_1, q_2)$ has to be placed in a new sub-circuit G_1, since it also acts on qubit q_1. The third sub-circuit G_3 contains two gates, i.e., $CNOT(q_1, q_0)$ and $CNOT(q_2, q_3)$, since they act on disjoint sets of qubits. Continuing this procedure results in $k = 10$ sub-circuits. That are indicated by dashed lines in Figure 1.*

In the following, we discuss how the A* search algorithm is used to determine the "best" permutation (with respect to a certain cost metric) in between two sub-circuits to be mapped. Note that no such call of the algorithm is required for the first sub-circuit since the effect of these SWAP gates are directly incorporated into the initial mapping of the logical qubits.

3.2 A* Search

How to conduct mapping of quantum circuits using A* search algorithms is described in two steps. First, we review how A* search works in general. Afterwards, its utilization in the considered problem is described.

General Algorithm The A* algorithm is a state-space search algorithm. To this end, (sub-)solutions of the considered problem are represented by state nodes. Nodes that represent a solution are called *goal nodes* (multiple goal nodes may exist). The main idea is to determine the cheapest path (i.e., the path with the lowest cost) from the root node to a goal node. Since the search space is typically exponential, sophisticated mechanisms are employed in order to consider as few paths as possible.

[6] Note that 1-qubit gates can be neglected when forming the sub-circuits.

[7] Note that a similar strategy is used in [24] (even though the permutation is not found using A* search).

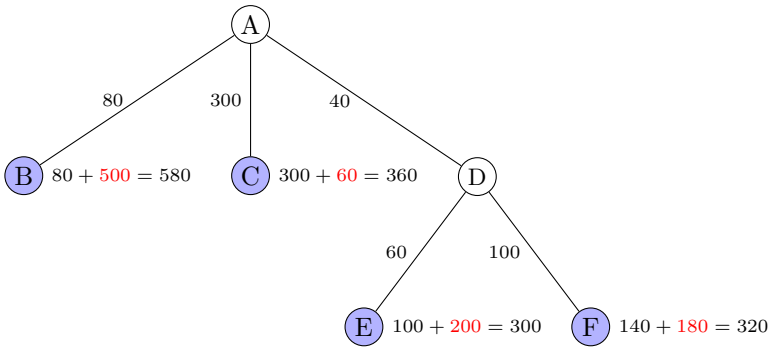

Fig. 4. A* search algorithm

All state-space search algorithms are similar in the way that they start with a root node (representing an initial state) which is iteratively expanded towards a goal node (i.e., one of the desired solutions). How to choose the node that shall be expanded next depends on the actual search algorithm. For A* search, we determine the cost of each leaf-node of the search tree. Then, the node with the lowest cost is chosen to be expanded next. The cost of a node x is given by $f(x) = g(x) + h(x)$. The first part, $g(x)$, describes the *path cost* of the current state (i.e., the cost of the path from the root to x). The second part provides an approximation of the remaining cost (i.e., the path cost from x to a goal node), which is estimated by a *heuristic cost* function $h(x)$. Since the node with the lowest cost is expanded, some parts of the search space (those leading to expensive solutions) are never expanded.

Example 5 *Consider the search tree shown in Figure 4. This tree represents the part of the search space that has already been explored for a certain search problem. The nodes that are candidates to be expanded in the next iteration of the A* algorithm are highlighted in blue. For all these nodes, we determine the cost* $f(x) = g(x) + h(x)$*. This sum is composed by the cost of the path cost from the root to* x *(i.e., the sum of the cost annotated at the respective edges) and the estimated path cost from* x *to a goal node (highlighted in red). Consider the node labeled* E*. This node has cost* $f(E) = (40 + 60) + 200 = 300$*. The other candidates labeled* B*,* C*, and* F *have cost* $f(B) = 580$*,* $f(C) = 360$*, and* $f(F) = 320$*, respectively. Since the node labeled* E *has the fewest expected cost, it is expanded next.*

Obviously, the heuristic cost should be as accurate as possible, to expand as few nodes as possible. If $h(x)$ always provides the correct minimal remaining cost, only the nodes along the cheapest path from the root node to a goal node would be expanded. But since the minimal costs are usually not known (otherwise, the search problem would be trivial to solve), estimations are employed. However, to ensure an optimal solution, $h(x)$ has to be *admissible*, i.e., $h(x)$ must not overestimate the cost of the cheapest path from x to a goal node. This ensures that no goal node is expanded (which terminates the search algorithm) until all nodes that have the potential to lead to a cheaper solution are expanded.

Example 5 (continued) *Consider again the node labeled* E*. If* $h(x)$ *is admissible, the true cost of each path from this node to a goal node is greater than or equal to 200.*

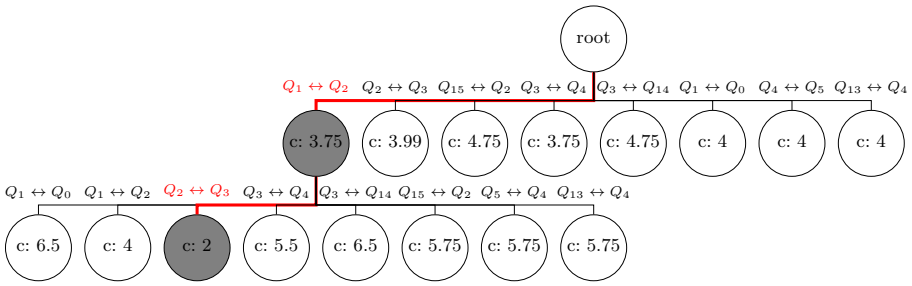

Fig. 5. First two expansion steps

Using A* for Mapping To utilize the A* algorithm recapitulated above for searching for the "best" permutation in between two sub-circuits of the circuit to be mapped, we have to define (1) the semantics of a node, (2) an expansion strategy for the nodes, and (3) a cost function to determine which node is expanded next.

Semantics of the nodes: Each node in the A* search adaption for the quantum-circuit mapping problem represents a mapping of the logical qubits of the quantum circuit to the physical ones of the quantum hardware. The root node for our search represents the mapping found by the last call of the search algorithm (or the initial mapping when searching for the permutation after the first sub-circuit G_0). Each node that represents a mapping that satisfies all coupling-constraints for the gates in the currently considered sub-circuit is a goal node.

Expansion strategy: As discussed in Section 2.2, the mapping is changed dynamically by inserting SWAP operations. Hence, a node is expanded by adding one node with a correspondingly modified mapping for each possible SWAP operation (according to the coupling map). To reduce the search space, we consider only SWAP operations that affect physical qubits to which a logical qubit is mapped that also occurs as control or target in a CNOT gate of the currently considered sub-circuit.

Example 6 *Considering again the quantum circuit shown in Figure 1 and assuming that the logical qubits q_0, q_1, q_2, q_3, and q_4 are mapped to the physical qubits Q_0, Q_1, Q_2, Q_3, and Q_4 (cf. Figure 2), respectively. A permutation has to be inserted before sub-circuit $G_8 = \{CNOT(q_2, q_3), CNOT(q_1, q_4)\}$ since the coupling-constraints are not satisfied for $CNOT(q_1, q_4)$—there is no arrow between Q_1 and Q_4 in the coupling map. A* search is applied to determine the best permutation circuit. Expanding the root node of the search tree (i.e., the node representing the current mapping) causes 8 successors (instead of 22) since there exist eight connections in the coupling map that affect the physical qubits Q_0, Q_1, Q_2, Q_3, or Q_4.*

Cost function: Eventually, we need to specify a cost function for the nodes in the search tree to determine which node has to be expanded next. For demonstration purposes, we describe the cost function for optimizing with respect to the overall number of additional gates in this section. In the following section, we show the flexibility of the A*-based approach by extending the cost function such that other objectives are supported. Recall that the cost function f of a node $f(x) = g(x) + h(x)$ is composed of the *path cost* as well as the *heuristic cost* that estimates the remaining cost for reaching a goal state.

The current cost of a node x is determined by its depth in the search tree since this describes the number of SWAP gates required to reach the mapping described by x. Since a SWAP gate is composed of 7 elementary operations (3 CNOTs and 4 Hadamard gates), the path cost could be defined by

$g(x) = 7 \cdot depth(x)$. However, in order to make this value better comparable to the cost functions of other objectives, just the number of SWAPs is used as cost, i.e., $g(x) = depth(x)$.

Usually, it is harder to find a good heuristic to estimate the remaining path cost for reaching a goal state since the heuristic shall be as accurate as possible (to prune large parts of the search space) while being admissible (i.e., not overestimating the true remaining cost) if an optimal/minimal solution is desired. To get an admissible heuristic, one has to determine the distance (i.e., the number of edges) of the logical qubits[8] in the coupling map for each CNOT gate in the currently considered sub-circuit, and take the maximum of all these distances.[9] However, since we do not aim for a locally optimal solution anyway (since this often affects the overall solution negatively [22]), we drop the admissibility constraint and specify the heuristic cost by accumulating all these distances.

Having $g(x)$ and $h(x)$ allows to utilize the A* search algorithm as introduced above. However, we can exploit the knowledge that it is called once for each sub-circuit G_i (except for the first one). Since we aim for a globally optimal solution rather than a local one, we additionally define *lookahead cost* $l(x, G_i)$ that estimates how the current mapping affects future sub-circuits. This term is added to the cost function, i.e., $f(x) = g(x) + h(x) + l(x, G_i)$. The lookahead cost contains an additive term for each subsequent sub-circuit G_j $(i < j < k)$ that is computed as sum of the distances of the target and control qubits of the CNOTs (like the heuristic cost). However, these additive terms are weighted with factors that decrease exponentially with $j - i$.

Example 6 (continued) *Figure 5 shows the search tree for finding the cheapest permutation circuit. The cost for the leftmost node (highlighted in gray) with depth 1 of the search tree has a path cost of* $g(x) = 1$ *since one SWAP operation (i.e.,* $Q_1 \leftrightarrow Q_2$*) has been added to reach this mapping. The heuristic cost is determined as follows: After the SWAP, the distance between the logical qubits* q_1 *and* q_4 *is 2 (since they are mapped to the physical qubits* Q_2 *and* Q_4*, respectively). Similarly, the distance between the logical qubits* q_2 *and* q_3 *is also 2. Hence,* $h(x) = (2-1) + (2-1) = 2$*. Since the control and the target qubit of the CNOT in the next sub-circuit (i.e.,* $G_9 = \{CNOT(q_2, q_3)\}$*) have also a distance of 2, the lookahead cost is* $l(x, G_8) = (2-1) \cdot 0.75 = 0.75$*) when using a weight of* 0.75*. Overall, this sums up to cost* $f(x) = 1 + 2 + 0.75 = 3.75$*. Similarly, the cost of the node highlighted in gray with depth 2 is* $f(x') = 2$*. Since this is a goal node, the new mapping is determined by inserting a permutation circuit composed of the SWAPS* $Q_1 \leftrightarrow Q_2$ *and* $Q_2 \leftrightarrow Q_3$*—eventually resulting in the mapped circuit shown in Figure 6. This circuit has a gate count of* $22 + 2 \cdot 7 = 36$*, a depth of 25, as well as a workload distribution of 28.*

3.3 Flexibility Regarding Different Objectives

Having the general scheme of A*-based quantum-circuit mapping as discussed above, we eventually can discuss the flexibility of this solution with respect to different architectures and objectives. As already stated in Section 2.2, the A* based approach is flexible regarding the architecture since coupling maps allow to specify not only IBM QX architectures, but also arbitrary ones like NN

[8] More precisely, the distance of the physical qubits to which the logical ones are mapped is taken.

[9] Note that the distance might also include 4 Hadamard gates to indicate that the direction of the CNOT has to be switched.

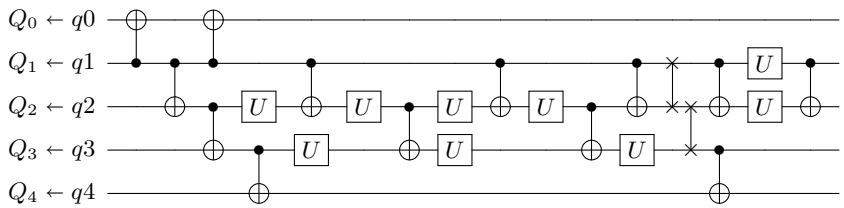

Fig. 6. Quantum circuit mapped to IBM QX5

architectures (since the distance of two physical qubits in the coupling map can be easily determined in linear time using Dijkstra's algorithm). Besides that, the decision how to group gates also allows for a large flexibility when using A*-based mapping. In this section, we demonstrate that the algorithm is also flexible regarding certain cost metrics by providing a generic and parameterizable cost function that can be extended to take other cost metrics into account (e.g., qubit fidelity [28]) in the future.

The *path cost* $g(x)$ of a node x is generalized in such a way that it does not only include cost resulting from the gate count of the permutation circuit $cost_g$ (i.e., the depth of x in the search tree), but also cost resulting from the circuit depth $cost_d$ as well as cost resulting from the workload distribution $cost_w$ of part of the circuit that is already mapped (including the permutation circuits described by x), i.e.,

$$g(x) = cost_g/7 \cdot w_0 + cost_d/5 \cdot (1 - w_0) + cost_w \cdot w_1. \tag{1}$$

Here, the additional weights w_0 and w_1 (with $0 \leq w_0, w_1 \leq 1$) allow to specify how much the algorithm shall focus on a certain cost metric.[10] For example, setting $w_0 = 1$ and $w_1 = 0$ results in the objective function described in the previous section—optimizing only the number of additional gates.

The heuristic cost (estimating the remaining cost based on the current mapping described by x) contains only the number of SWAPs to reach a goal node $cost_{hg}$ (this gives an estimate for gate count as well as for circuit depth):

$$h(x) = cost_{hg}/7 \tag{2}$$

Like the path cost, also the lookahead cost for a subsequent sub-circuit G_j ($i < j < k$) is generalized to a sum of three terms:

- The sum of distances of the qubits occurring in the CNOTs of the sub-circuit G_j, i.e., $cost_{lg}$,
- the increase of circuit depth based on the distance of the qubits in the occurring CNOTs, i.e., $cost_{ld}$, and
- the change in the workload distribution based on the distance of the qubits in the occurring CNOTs, i.e., $cost_{lw}$.

This leads to the generalized lookahead cost

$$l(x) = (cost_{lg}/7 \cdot w_0 + cost_{ld}/5 \cdot (1 - w_0) + cost_{lw} \cdot w_1) \cdot w_2^{j-i}. \tag{3}$$

Here, the additional weight w_2 allows to exponentially decrease the contribution of future sub-circuits.

[10] Note that we store the depth and the workload distribution for each physical qubit (considering the already mapped part of the circuit) to keep track of these values.

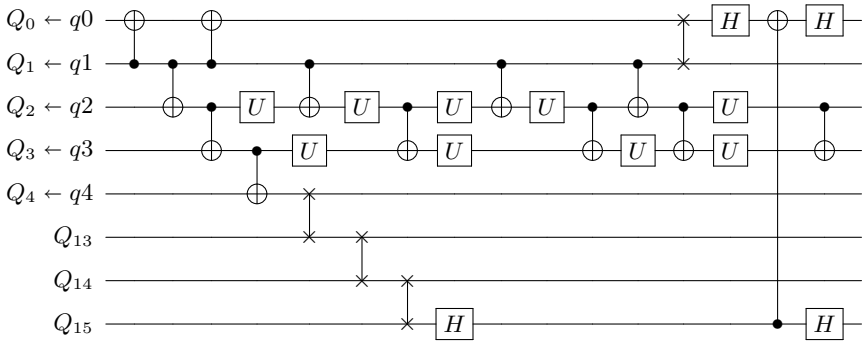

Fig. 7. Quantum circuit mapped to IBM QX5 using depth optimization

Overall, this leads to the generalized cost function

$$f(x) = g(x) + h(x) + \sum_{j=i+1}^{k} l(x, G_j) \tag{4}$$

for a node x when currently considering a sub-circuit G_i.

Example 6 (continued) *Setting weights of the generalized cost and heuristic functions to* $w_0 = 0.05$*,* $w_1 = 0$*,* $w_2 = 0.75$ *allows to optimize for circuit depth at first hand and not for the gate count. Using this cost function in the mapping algorithm results in the circuit shown in Figure 7. This circuit has now a depth of 22 (instead of 25) at the cost of increasing the gate count from* 36 *to* $26+4 \cdot 7 = 54$*.*

4 Experimental Evaluation

In this section, we experimentally evaluate the flexibility of A*-based mapping. To this end, we compare the generic solution proposed in this paper to dedicated solutions for 1D NN architectures developed over the past 10 years. Moreover, we evaluate how the parameters of the generalized objective function affect the cost of the mapped circuits. To this end, we have implemented the proposed general mapping approach in C++ and conducted several evaluations using benchmarks from RevLib [31] on a laptop with 2.6 GHz and 4 GB RAM.[11]

4.1 Flexibility Regarding the Considered Architecture

In a first series of evaluations, we compare the proposed generic mapping algorithm to dedicated solution for 1D NN architectures. As discussed in Section 2.2, these architectures can be modeled easily by using coupling maps, but are more restricted which makes it easier to develop dedicated optimizations. We compare the proposed approach to one of the first methods developed for these kind of architectures [17] as well as to one of the latest and most elaborated solutions [19] (this way, we showcase, how an adapted A*-based version compares to the initial NN-methods as well as today's state-of-the-art methods that emerged after several years of research on nearest neighbor optimization). Since both try

[11] Note that we grouped all gates that act on disjoint qubits into a sub-circuit as done in [21, 22] (neglecting 1-qubit gates when forming the sub-circuits).

Table 1. Comparison to dedicated solutions for 1D NN architectures

Benchmark			Required SWAP operations			Improvements	
Name	n	$\|G\|$	[17]	[19]	proposed	w.r.t. [17]	w.r.t. [19]
3-17	3	13	5	6	4	-1	-2
4gt10-v1	5	36	29	24	22	-7	-2
aj-e11	5	59	43	33	29	-14	-4
hwb5	5	106	86	66	59	-27	-7
hwb6	6	146	140	111	104	-36	-7
mod5adder	6	81	79	46	54	-25	8
ham7	7	87	86	72	71	-15	-1
QFT7	7	21	29	18	21	-8	3
rd53	7	78	96	66	61	-35	-5
hwb7	8	2659	3480	2067	2015	-1465	-52
QFT8	8	28	41	31	34	-7	3
urf2	8	25150	23608	18428	16597	-7011	-1831
hwb8	9	16608	21767	13176	13546	-8221	370
QFT9	9	36	66	49	47	-19	-2
urf1	9	57770	62019	45730	42219	-19800	-3511
urf5	9	51380	54038	39852	37066	-16972	-2786
hwb9	10	20405	32979	18988	19495	-13484	507
QFT10	10	45	96	64	61	-35	-3
Shor3	10	2076	3353	2112	1982	-1371	-130
sym9	10	4452	5353	3103	4049	-1304	946
urf3	10	132340	140908	108321	100345	-40563	-7976
cycle10_2	12	1212	2193	966	1176	-1017	210
Shor4	12	10004	9510	5616	5410	-4100	-206
plus63mod4096	13	29019	54999	25617	29108	-25891	3491
plus127mod8192	14	65455	136820	63354	69470	-67350	6116
plus63mod8192	14	37101	77753	35472	38713	-39040	3241
Shor5	14	20530	22846	12221	11302	-11544	-919
ham15	15	458	803	531	537	-266	6
urf6	15	53700	91563	54815	51666	-39897	-3149
Shor6	16	37770	41551	22829	21159	-20392	-1670

to minimize the number of additional SWAP operations,[12] we also set our parameters $w_0 = 1$, $w_1 = 0$, and $w_2 = 0.75$ for $j - i = 1$ and to $w_2 = 0.5$ for $j > i + 1$.

Table 1 summarizes the obtained results when using the proposed approach for mapping all benchmark listed in the respective papers. The first three columns list the name of the benchmark, the number of qubits n, as well as the number of gates in the circuit to be mapped $|G|$. The next three columns list the obtained number of additional SWAP operations for the dedicated solutions presented in [17] and [19] as well as for the generic approach presented in this paper. The last two columns list the respectively achieved improvements. Runtimes are not provided since all mappings have been determined within a couple of seconds.

As can be seen in Table 1, the proposed generic approach significantly outperforms the dedicated approach presented in [17]. On average, 35.7% fewer SWAP operations are inserted. The generic approach even provides similarly good results compared to one of the most elaborated approaches for these specific architectures [19]. On average, the number of additionally required SWAP operations reduces even by 1.4%. These results are rather remarkable, since they indicate that dedicated solutions for, e.g., 1D NN architectures do not perform better than generic solutions applicable to any kind of envisioned architecture.

Overall, our evaluation shows that we reach significant and minor improvements compared to [17] and [19], respectively, even though the generality and flexibility of our approach does not allow to utilize dedicated optimization techniques when mapping to NN architectures. This is a clear testament of the power of A* as it shows that, using the proposed method allows to determine much better results as initial version and very competitive results compared to recent dedicated solutions.

[12] Note that no Hadamard operations have to be inserted since these architectures allow CNOTs in any direction between neighboring qubits.

Table 2. Evaluation of different parameter settings

Benchmark			Opt. gate count			Opt. depth			Opt. workload dist.		
Name	n	$\|G\|$	gc	cd	wd	gc	cd	wd	gc	cd	wd
3-17	3	36	105	63	14	112	64	9	138	74	10
rd32-v1	4	36	112	66	51	120	65	32	127	68	34
4_49	5	217	706	408	204	733	376	282	805	471	210
4gt10-v1	5	148	513	294	174	524	253	199	499	293	151
ex3	6	403	1345	762	447	1463	685	408	1338	768	333
hwb5	6	1336	4319	2512	1550	4689	2284	1666	4457	2547	1488
4mod5-bdd	7	70	270	151	116	285	130	87	274	145	98
ham7	7	320	1136	660	464	1287	609	392	1407	811	317
cm82a	8	650	2284	1242	531	2660	1131	697	2618	1454	630
urf5	9	164416	532100	294007	150186	612474	285427	163635	615963	332710	172780
sqn	10	10223	35572	19534	7354	41422	17673	10533	35860	19538	7144
urf3	10	125362	459017	250539	117506	560534	234612	144637	619623	324165	111370
9symml	11	34881	120708	67381	23653	149082	62038	33838	122708	68333	22098
dc1	11	1914	6841	3756	1264	7691	3337	1717	6369	3505	1466
life	11	22445	78395	44236	14841	95975	40431	21876	78548	44172	13975
rd84	12	13658	47471	25604	9203	58654	23878	12679	47833	25737	9101
sqrt8	12	3009	10910	5979	1906	12836	5352	2667	10923	6021	1917
sym10	12	64283	225510	126852	42949	278390	115416	60660	260146	146408	39162
adr4	13	3439	11569	6259	2616	14086	5990	3401	11878	6386	2360
dist	13	38046	133149	70751	22399	163240	64447	33983	133160	70283	22303
squar5	13	1993	6984	3630	1483	8282	3397	1940	7177	3678	1494
0410184	14	211	864	425	156	1147	347	184	891	452	107
pm1	14	1776	5971	3149	1454	6985	3046	1865	6451	3426	1110
sao2	14	38577	136483	71152	21215	169801	65420	35041	137215	71131	20456
co14	15	17936	64910	31600	6403	85351	29466	12829	70202	33803	5381
square_root	15	7630	26733	14022	4274	31663	12373	5895	27274	14244	3731
urf6	15	171840	617067	324396	88294	727960	307340	132278	628436	329294	74676
cnt3-5	16	175	555	204	44	674	224	79	579	220	49
inc	16	10619	36524	20579	6551	44182	18813	9890	37521	21126	4859
mlp4	16	18852	67821	37902	10773	83733	33734	15466	68415	37865	8728

4.2 Flexibility Regarding Different Cost Metrics

In a second round of experiments, we analyze the flexibility of the proposed method with respect to different objectives. In fact, changing a few parameters allows to optimize for different cost metrics without changing any code or developing a new and dedicated solution.

Table 2 summarizes the obtained results. The first three columns list the name of the benchmark, the number of qubits n, the depth of the gate count of circuit to be mapped $|G|$. In the remaining columns, we list the gate count of the mapped circuit *gc*, its depth *cd*, as well as the workload distribution of the qubits *wd* for the proposed circuit when optimizing for gate count (by setting $w_0 = 1$, $w_1 = 0$), for circuit depth (by setting $w_0 = 0.04$, $w_1 = 0$), and for the workload distribution (by setting $w_0 = 1$, $w_1 = 0.1$) when mapping the circuits to IBM QX5 (w_2 was the same for all three mappings and was 0.75 for $j - i = 1$ and 0.5 for $j > i + 1$).

Considering the optimization with respect to gate count as baseline, changing the parameters for optimizing with respect to circuit depth indeed results in a decrease of depth (on average by 7.7%) at the expense of inserting more SWAP and Hadamard operations (on average 17.1%). Similarly, the depth of the circuits optimized for depth is on average 12.3% smaller compared to those optimized for the workload distribution of the qubits. In contrast, their workload distribution is 39.3% worse compared to circuits optimized for that cost metric.

Overall, the experimental evaluation confirms the flexibility of the proposed approach with respect to different objectives. Moreover, this shows that optimizing for different objectives or architectures does not require to develop new algorithms, but only to adjust very few parameters in the objective functions. By this, we provide a mapping solution which is inherently applicable for future architectures just by employing suitable parameters or by slightly modifying the cost function.

5 Conclusions

In this work, we evaluated the flexibility of A* for mapping quantum circuit to physical quantum computers. By using coupling maps to model restrictions in the qubit interactions of these devices, one can specify arbitrary quantum architectures (e.g., NN architectures or IBM QX architectures). We additionally provide a generic and parameterizable cost function, our approach allows to optimize for different design objectives (like gate count, circuit depth, or workload distribution) just by changing parameters and without writing any code—inherently providing a customized mapping algorithm. Our experimental evaluation shows, that this generic approach is competitive with dedicated approaches for NN architectures and that changing the parameters indeed significantly influence the design objectives as desired.

Acknowledgements

This work has partially been supported by the LIT Secure and Correct System Lab funded by the State of Upper Austria and the European Union through the COST Action IC1405.

References

1. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information. Cambridge Univ. Press (2000)
2. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Jour. of Comp. **26**(5) (1997) 1484–1509
3. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Symp. on Theory of Computing. (1996) 212–219
4. Montanaro, A.: Quantum algorithms: an overview. npj Quantum Information **2** (2016) 15023
5. Debnath, S., Linke, N., Figgatt, C., Landsman, K., Wright, K., Monroe, C.: Demonstration of a small programmable quantum computer with atomic qubits. Nature **536**(7614) (2016) 63–66
6. Linke, N.M., Maslov, D., Roetteler, M., Debnath, S., Figgatt, C., Landsman, K.A., Wright, K., Monroe, C.: Experimental comparison of two quantum computing architectures. Proceedings of the National Academy of Sciences (2017) 201618020
7. IBM Q team: IBM Q. `https://www.research.ibm.com/ibm-q/` Accessed: 2019-02-05.
8. Nay, C.: IBM unveils world's first integrated quantum computing system for commercial use. `https://newsroom.ibm.com/2019-01-08-IBM-Unveils-Worlds-First-Integrated-Quantum-Computing-System-for-Commercial-Use` Accessed: 2019-02-05.
9. Preskill, J.: Quantum computing in the NISQ era and beyond. arXiv preprint arXiv:1801.00862 (2018)
10. Sete, E.A., Zeng, W.J., Rigetti, C.T.: A functional architecture for scalable quantum computing. In: Int'l. Conf. on Rebooting Computing. (2016) 1–6
11. Neill, C., Roushan, P., Kechedzhi, K., Boixo, S., Isakov, S.V., Smelyanskiy, V., Megrant, A., Chiaro, B., Dunsworth, A., Arya, K., et al.: A blueprint for demonstrating quantum supremacy with superconducting qubits. Science **360**(6385) (2018) 195–199
12. Barenco, A., Bennett, C.H., Cleve, R., DiVincenzo, D.P., Margolus, N., Shor, P., Sleator, T., Smolin, J.A., Weinfurter, H.: Elementary gates for quantum computation. Physical review A **52**(5) (1995) 3457
13. Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. Transactions on Computer-Aided Design of Integrated Circuits and Systems **32**(6) (2013) 818–830

14. Miller, D.M., Wille, R., Sasanian, Z.: Elementary quantum gate realizations for multiple-control toffoli gates. In: Int'l Symp. on Multi-Valued Logic. (2011) 288–293
15. Siraichi, M., Dos Santos, V.F., Collange, S., Pereira, F.M.Q.: Qubit allocation. In: International Symposium on Code Generation and Optimization. (2018) 1–12
16. Botea, A., Kishimoto, A., Marinescu, R.: On the complexity of quantum circuit compilation. In: Symposium on Combinatorial Search. (2018)
17. Saeedi, M., Wille, R., Drechsler, R.: Synthesis of quantum circuits for linear nearest neighbor architectures. Quantum Information Processing **10**(3) (2011) 355–377
18. Wille, R., Lye, A., Drechsler, R.: Exact reordering of circuit lines for nearest neighbor quantum architectures. Transactions on Computer-Aided Design of Integrated Circuits and Systems **33**(12) (2014) 1818–1831
19. Wille, R., Keszocze, O., Walter, M., Rohrs, P., Chattopadhyay, A., Drechsler, R.: Look-ahead schemes for nearest neighbor optimization of 1D and 2D quantum circuits. In: Asia and South Pacific Design Automation Conference. (2016) 292–297
20. Hattori, W., Yamashita, S.: Quantum circuit optimization by changing the gate order for 2D nearest neighbor architectures. In: International Conference on Reversible Computation, Springer (2018) 228–243
21. IBM Q team: QISKit Python SDK version 0.4.15. `https://github.com/QISKit/qiskit-sdk-py` Accessed: 2019-05-02.
22. Zulehner, A., Paler, A., Wille, R.: An efficient methodology for mapping quantum circuits to the IBM QX architectures. Transactions on Computer-Aided Design of Integrated Circuits and Systems (2018)
23. Zulehner, A., Wille, R.: Compiling SU(4) quantum circuits to IBM QX architectures. In: Asia and South Pacific Design Automation Conference. (2019) 185–190
24. Itoko, T., Raymond, R., Imamichi, T., Matsuo, A., Cross, A.W.: Quantum circuit compilers using gate commutation rules. In: Proceedings of the 24th Asia and South Pacific Design Automation Conference. (2019) 191–196
25. Dueck, G.W., Pathak, A., Rahman, M.M., Shukla, A., Banerjee, A.: Optimization of circuits for IBM's five-qubit quantum computers. In: Euromicro Conference on Digital System Design. (2018) 680–684
26. Wille, R., Burgholzer, L., Zulehner, A.: Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations. In: Design Automation Conference. (2019)
27. Zulehner, A., Wille, R.: Advanced simulation of quantum computations. Transactions on Computer-Aided Design of Integrated Circuits and Systems (2018)
28. Tannu, S.S., Qureshi, M.K.: Not all qubits are created equal: A case for variability-aware policies for NISQ-era quantum computers. In: International Conference on Architectural Support for Programming Languages and Operating Systems. (2019) 987–999
29. Kelly, J.: A preview of Bristlecone, Google's new quantum processor. `https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html` (2018)
30. IBM Q team: IBM Q 16 Rueschlikon backend specification v1.1.0. `https://ibm.biz/qiskit-rueschlikon` Accessed: 2019-02-05.
31. Wille, R., Große, D., Teuber, L., Dueck, G.W., Drechsler, R.: RevLib: an online resource for reversible functions and reversible circuits. In: Int'l Symp. on Multi-Valued Logic. (2008) 220–225 RevLib is available at http://www.revlib.org.