Near Optimal Fidelity in Quantum Circuits through Incorporating Efficient Real-time Error Based Heuristics in Compiler Mappings

Md. Nurul Muttakin
Bangladesh University of Engineering and Technology

Abstract
To run a quantum program in the real device, compiler maps the logical qubits to physical qubits. This is the most crucial step of compiling a quantum circuit. Because the fidelity of a quantum circuit depends heavily on this mapping process. However, this qubit mapping problem is NP-complete. Therefore, we should resort to heuristics to find high fidelity mappings.

In this paper, we focused on finding efficient heuristic techniques to incorporate real-time error feedback and device connectivity information in order to achieve high fidelity mapping of quantum circuit. We performed extensive analysis and experimental study based on two baseline algorithms. We performed our experimentation on various combination of different error rates and heuristic techniques. Consequently, we designed very elegant techniques to consider both all type of real-time error feedback, and connectivity information. We showed that our best heuristic approach performs $1.62x$ (on average) better than one baseline and $1.934x$ (on average) better than the other baseline on random benchmarks. Finally, we compared our best heuristic (CAES) with the state of the art heuristic based mapping algorithm on representative benchmarks. We found that CAES performed $1.7x$ (on average) better than the state of the art in terms of success rate.

1 Introduction:
We want to use the immense computing power of nature using quantum computing (QC) [7]. Some of the appealing applications of QC systems are: quantum simulations of chemical compounds for new drug discovery [14, 22]; great improvement in artificial intelligence, specially in machine learning [25]; drastic change in the whole cryptography or communication technology [29]. However, most of the existing QC systems have less than hundreds qubits and are error prone. The quantum computers having 10 to 1000 qubits are termed as Noisy Intermediate Scale Quantum (NISQ) computers [24]. There are applications where these NISQ devices can provide benefits over classical computers [19]. Developing software systems for this NISQ-era QC systems is an urgent need. Specially, compilers which incorporate the real time error feedback from the real device in order to provide higher fidelity. One of the the most important functions of a quantum compiler is mapping. When a programmer writes a quantum program without considering the underlying hardware, the task of compiler is to translate the program suitably for the underlying hardware. Compiler has to map the program qubits (logical qubits) to physical qubits. The mapping process must conform the hardware constraints of the real device. These constraints arise from the connectivity of the qubits of real device. To execute a multi-qubit gate, for example a CNOT gate, in the real hardware there must be a direct connection between the two qubits of the CNOT gate. If the direct connection is not possible, then qubits must be taken closer using swap gates. As the device is error prone, a mapper should consider error feedback in order to get high fidelity outputs. This qubit mapping problem is proved NP-complete [30]. As a result, the optimal solution considering real-time error feedback is not scalable for more than 32 qubits [20]. Therefore, it is utmost necessary to find heuristics based approach to address the problem. These heuristics should consider the real-time error feedback as well as connectivity information of the device. However, an extensive study on the heuristic techniques to incorporate real-time error data, and connectivity information does not exist in the literature.

In our work, we performed extensive experimental study as well as theoretical analysis to find effective heuristic techniques. In our various heuristics, we incorporated real-time error feedback based on two baseline algorithms. We invented elegant techniques in order to consider both of real-time error feedback, and connectivity of information of qubits simultaneously. Our main objective is the fidelity enhancement using heuristic techniques incorporating real time error feedback. To this end, we performed experimentation based on the baseline algorithms. We found that our best heuristic method...
performs 1.62x (on average) better than one baseline [40] and 1.934x (on average) better than the other baseline [41]. Moreover, the second baseline algorithm won IBM’s qiskit developer challenge [41]. Finally, we compared our best connectivity aware error sensitive (CAES) heuristic with the state of the art heuristic based mapping algorithm [20]. Our CAES heuristic performed 1.7x better than the state of the art in terms of fidelity. Our key contributions are summarized as follows:

- Extensive experimentation on real device in order to find effective (high fidelity) methods to incorporate real time error feedback in compiler mappings.
- Invention of elegant techniques to incorporate real time error feedback and connectivity of the device simultaneously in order to get near optimal fidelity
- Discovering a connectivity aware error sensitive (CAES) mapping algorithm using our invented heuristics which performs 1.7x (on average) better than the state of the art heuristic based mapping algorithm.
- Developing criteria for designing heuristic based mapping algorithm in order to achieve high fidelity.

2 Background:

In this section we are going to provide a basic idea on quantum computing related to our paper.

**Qubits:** Qubits of a quantum computer are a little analogous to the bits of classical computer. A qubit has two basis states analogous to 0 and 1 of classical bits. However, there are substantial differences between bits and qubits. A qubit can be in any linear combination of the basis states, called superposition [35]. Additionally, two or more qubits can be very closely correlated, which is called entanglement [37]. To entangle two qubits there must have direct hardware connection in real device between the qubits. In quantum device, the state of qubit never can be perfectly copied according to the no cloning theorem of quantum mechanics [36].

**Quantum Operation:** Quantum operations are performed by manipulating the qubits. There are two types of basic quantum operations: application of gates and measurement operation. If a quantum operation involves only a single qubit, then it is termed as single qubit gate. If it involves multiple qubits, then it is called multiple qubit gate. Operations involving multiple qubits need entanglement [38]. Currently available multiple qubit gate in IBM’s online available quantum computers is the CNOT gate [13]. Since entanglement is necessary for CNOT gate, there must be a physical hardware connection between associated qubits. In figure 1(a) both single qubit gates and CNOT gates are shown.

**Quantum Circuit:** Quantum circuit refers to the sequential arrangements of quantum logic gates in different qubits.

When a quantum circuit is drawn, each qubit is represented as a line. This line is a timeline. The gates are applied in the sequential order of time in this timeline. In figure 1(a) a quantum circuit is drawn.

**Layer:** A layer is a set of gates which can be applied in the real hardware in parallel. In figure 1(b) layers of a quantum circuit are shown. The idea of layer is provided in [40].

**Group:** A group is a minimal set of gates which act on the same set of qubits sequentially [41]. At the current status of the device, a group is associated with a single qubit or two qubits. Different groups are shown in figure 7 1(c). All the groups of a quantum circuit form a DAG as shown in figure 1(d).

**Coupling map:** The so called coupling map actually the connectivity graph of the qubits in a quantum computer. In an ideal quantum computer each qubit must be directly connected to all other qubits. However, at the current stage of technology it is not possible to connect every qubit to every other qubits. As a result, there are limited connectivity. The connectivity is depicted by coupling map. The coupling map of IBM’s quantum computer at Melbourne is shown in figure 4.

**Real time error parameters:** The real time error parameters we consider here, are based on real time error rates of different qubits and links. For qubits, we consider single qubit gate error rates and readout error rates. For links between qubits, we consider CNOT error rates. These error rates are available online for IBM’s online quantum computers [13]. We do not consider parameters related to decoherence time. Because these parameters have negligible effect on the small circuits [20]. Experimentation with large circuits is not possible at the current status of QC systems.

3 Related Work:

None of the work in the literature performed extensive experimental study to incorporate real-time errors or noise in order to improve fidelity using heuristic based algorithm. Most of the works considered these real-time errors based on some intuition. Experimental study in the real quantum devices, like IBM’s quantum experience, are very few. Additionally the optimal approach provided by [20] is not scalable for more than 32 qubits. Therefore to develop effective, scalable, real-time error sensitive mapping algorithms for NISQ devices, we must resort to heuristic based approaches. The authors of [20] also provided two heuristic based approaches, where the edge based heuristic performed better than the vertex based heuristic. However, the connectivity information of different qubits and different edges are not considered. Single qubit gate errors are also not considered. The edge based heuristic considers the the edge first which has the highest CNOT and readout reliability. However, the best edge having highest CNOT and readout reliability may have the worst connectivity [4]. As a result, many swap operations will be required, which will ultimately reduce performance greatly. Their reliability met-
The sum of negative log of errors over path a-b-c is 1+1=2 and over path a-d-c is 2+2=4. This value is minimized over path a-b-c. However, this path is not the most reliable one, because it has the higher error prone links. On the other hand, in path a-d-c the links are less error prone. Simply minimizing sum of errors will take the more reliable path a-d-c.

The edge weight was taken as the negative log of CNOT error rate. Then dijkstra’s algorithm was run to find the most reliable path between two qubits. The dijkstra’s algorithm is a shortest path algorithm [5]. Therefore the most reliable path minimizes the sum negative log error along the path, which basically maximizes the multiplication of error rates of each link along the path. As a result the path can’t never be the most reliable path. This is explained in figure 2. [32] incorporated errors in [40], which is our first baseline. Vertex based heuristics provided in [32]. The strength of a vertex is taken as the sum of error rates for all the links connected to that vertex. If this value is high, then the sum of error rates is high in that vertex which is not desirable. This can be verified in figure 2, where we can assume for simplicity all the qubits has the same readout and single qubit gate error rates. Therefore, according to [32], the strongest node is e. However, this node has highest CNOT error rates. Actually, the strongest node will be node c. Because it has similar connectivity as node e but error rates are much lower than e. Therefore, this heuristic is not a good heuristic. In fact, it is desirable that the error rates should be lower as well as connectivity would be higher. Also the edge based heuristics are better than vertex based heuristics [20]. The authors of [8] considered only error rates of CNOT gates. Readout error rates, single qubit gate error rates were not considered. Moreover, connectivity of qubits is not considered, although connectivity is very important. Finally, no reasonable experimental results were shown on real quantum devices in [8]. The importance of errors of real quantum device also showed in [33]. All other works related to qubit mapping did not consider the real-time noise data or error rates. Authors of [11] considered equal gate duration for all gates to find heuristic based scheduling of quantum circuits in linear coupling map or qubit connectivity graph. The approaches provided in [30] performed compilation of small circuits in IBM’s quantum devices without considering real time error rates. Authors of [39] also developed heuristics based approach for compiler mapping without any considerations of real time error data. In our baseline algorithms provided by [40, 41] considered the minimization of additional swap operations, i.e., qubit movement operations without considering real devices’ calibration data. Real-time noise or errors are not considered in [2-4, 6, 10, 15-17, 21, 23, 26–28, 34] also. Moreover, these works are not for general quantum programs or general quantum computing architecture, instead limited to specific quantum algorithms or specific architecture.

In this paper, we mainly focused on finding effective heuristic techniques to incorporate real-time error feedback in order to find high fidelity mapping of quantum circuit. We experimented on various combination of different error rates and analytical heuristics on real device like IBM’s quantum experience. We considered all type of error rates that is available online such as: single qubit gate error rates, multi-qubit gate error rates, readout error rates. Various well analyzed heuristics such as: error sensitive edge selection strategy, connectivity aware error sensitive strategy, group ordering in group based approach etc., and their various combinations also considered. To get unbiased estimation of performance of different approaches, we used random circuits in our experiment. Performing equal number of experiments on each variation in real device, we found that some of our approaches, in which connectivity is considered with error rates, get 1.934x (on average) improvement over [41]. In fact, none of the existing
work conducted extensive experimental study to find efficient heuristic techniques based on real device performance to incorporate real-time error feedback and qubits connectivity information in order to achieve high fidelity mapping. In our paper we consider the edge based heuristic in [20] as the state of the art heuristic based noise sensitive qubit mapping algorithm, because it has higher experimental validation and performance compared to its previous works and it the most recent work. We compared our best heuristic based mapping algorithm with the state of the art on representative benchmarks. We found that our approach gets 1.7x (on average) better performance in terms of fidelity or success rate. Since, the work in [20] did not consider qubits connectivity information in their heuristic properly, it gets the worst performance in the highly connected circuits.

4 Our Methodology:

We investigated various methods of incorporating different real-time errors of real quantum computers like IBM’s online devices in order to find high fidelity mappings. To this end, we used two baseline algorithms given by [40, 41]. These baseline algorithms did not consider the real-time error feedback. In this section we will describe the baseline algorithms in short detail; different approaches to incorporate different real-time errors and different heuristic techniques; and finally, the various combinations of applying different error incorporating techniques and heuristics to the baseline algorithms. Both of the baseline algorithms use A* search algorithm. The A* algorithm [12] is graph traversal algorithm to find cheapest path from a root node to some goal node. The algorithm can be used to search state space. An initial partial solution is used as root node. Starting at the root node the algorithm continuously expands other nodes recursively. Each new node expanded from other node, represents a more complete solution. Finally, the goal node represents the desired complete solution. The purpose of the algorithm is to find the cheapest path to some goal node. At each stage of the algorithm, it chooses the cheapest node to expand according to some cost function. The cost of node $x$ is $f(x) = g(x) + h(x)$. Here, $g(x)$ is the fixed cost to reach current node. The $h(x)$ is a heuristic estimate of cost to reach the goal node (desired solution) from the current node. There are two main parts of A* algorithm: one is the cost function and other is expansion strategy from a node (sub-solution) to children nodes (new more complete sub-solutions).

4.1 Baseline Algorithms:

We used two baseline algorithms to incorporate different real-time error rates and heuristics techniques. In these algorithms, error parameters are not considered [40, 41]. Initially both algorithms start with a void mapping. Void mapping means no program qubit (logical qubit) is mapped to any physical qubit. Therefore, both algorithms find a initial mapping for the circuit. All the subsequent gates are applied according to the initial mapping. However, some of the CNOT gates may not be applied according to the current mapping. In that case, both baselines use A* algorithm to find effective new mapping. The new mapping is achieved by applying swap operations. Therefore, the goal of the A* algorithm is to find optimal swap operations. The A* algorithm can provide optimal results if the heuristic function $h(x)$ is admissible. Admissible means $h(x)$ never overestimates the actual cost. However, admissible heuristics are often costly to calculate [40]. Therefore, the both baseline algorithm uses non-admissible heuristics. There are two main parts in both algorithms: one is the cost function of A* algorithm and other is the swap application strategy (expansion strategy of A* algorithm). The same cost function and swap application strategy can be applied in both algorithms. The swap application strategy means, which edge of the physical device to choose to apply a swap gate. In this case, the edge which has potential to reduce overall cost, should be chosen earlier. Baseline 1 fixes mapping layer by layer using A* search [40]. Baseline 2 algorithm performs mapping task group wise [41].
**Algorithm 1 (Baseline 1): Layer Based Approach**

1: divide quantum circuit into layers
2: for each CNOT gate in first layer do
3:   if both logical qubits are not mapped then
4:       take an unassigned edge from coupling map
5:       map both logical qubits to the edge
6:       apply the CNOT gate
7:   if one logical qubit ($lq$) is not mapped then
8:       find physical qubit ($pq$) of mapped logical qubit
9:       take the nearest physical qubit ($pq$) of $pq$
10:      assign $pq$ to $lq$
11:     try to apply the CNOT gate
12: for each layer do
13:   for each CNOT gate do
14:     if CNOT gate can be applied then
15:         apply the CNOT gate
16:     else
17:         run A* to find necessary swaps
18:         apply the CNOT gate

**Algorithm 2 (Baseline 2): Group Based Approach**

1: divide quantum circuit into groups
2: formulate DAG of groups
3: topologically sort the DAG
4: for each two qubit group in topological order do
5:   if both logical qubits are not mapped then
6:     take an unassigned edge from coupling map
7:     map both logical qubits to the edge
8:   if one logical qubit ($lq$) is not mapped then
9:     find physical qubit ($pq$) of mapped logical qubit
10:    take the nearest physical qubit ($pq$) of $pq$
11:    assign $pq$ to $lq$
12:   if assigned physical qubits are not adjacent then
13:     run A* algorithm to find necessary swaps
14:    apply all the gates in the group.
15: map single qubit groups to unassigned qubits.

We can observe that both of the algorithms uses A* search in the same way. Whenever any CNOT gate can not be applied , then A* algorithm finds the new mapping such that at least one more CNOT gate can be applied. In fact, A* algorithm returns the necessary swap gates to achieve the new mapping. Now we can explain how A* algorithm works in each cases. In the case of cost function, fixed cost of a node $x$, $g(x)$ is the sum of already used swap gate costs. Each swap gate cost is taken 7 [40]. As each swap gate needs three CNOT gate and four Hadamard ($H$) gates. In fact, A* algorithm starts with a given mapping. A* returns a new mapping with necessary swaps to achieve the new mapping from the given initial mapping. The criteria for goal node is the capability to apply at least one new CNOT gate. The heuristic cost functions $h(x)$ are different in each baseline. In the baseline 1, $h(x)$ is the sum of CNOT gate costs in the current layer and in the immediately next layer. In this case each CNOT gate cost is taken as 1. In case of baseline 2, for every pair of logical qubits mapped to physical qubits, the distance between the physical qubits is calculated considering each edge cost 1. Now when A* finds mapping for a group, then all the immediately next groups according to the DAG are considered to find $h(x)$. In fact, $h(x)$ is the sum of distances of all CNOT gates in the considered groups.

The expansion strategy is similar in both algorithms. Application of swap gate on some edge of physical device changes one mapping to another mapping. Therefore, successor mappings are achieved by applying all possible swap gates. Each application of swap gate on some edge creates a new mapping. Therefore, total number of edges in coupling map are the possible swap gates that can be applied in a node. Both algorithms expand to a successor by applying a swap gate on some edge. The edges are chosen in a natural order. The A* search process ends when it finds a mapping which allows the required CNOT gate to be applied.

The cost function in both algorithms are static. No error feedback from the real device are incorporated in the cost functions. Therefore, there are opportunities to incorporate real-time error feedback to the cost function. In the rest of the work, we consider how to incorporate different real time error feedback in the cost function. Moreover, we also consider how to incorporate single qubit gate cost.

Expansion strategy in the A* algorithm does not consider any edge ordering. Each edge in the real device has different error rates and connectivity. Some edges have better error rates than others. Therefore, good edges should be chosen earlier for expansion. Because the earlier it is chosen, the more it would be used in the circuit. As a result fidelity will be increased. To this end, we should have strategy to determine edge ordering i.e., the order in which different edges from the coupling map should be chosen for initial mapping process or expansion process. In the case of initial mapping, edges are needed to apply CNOT gates. On the other hand, in case of expansion process edges are needed to apply swap gate to get new mapping.

In spite of having better error rate, an edge may have worse connectivity. For example in figure 4, we can observe that edge (0, 14) has lower error rate but it has worse connectivity. Because both qubits are connected to only one other qubit. If this edge is chosen earlier, number of swaps will be increased. On the other hand the edge(1, 13) has greater error rate, at the same time has higher connectivity. The edge will reduce number of additional swaps. Therefore, in the case of edge selection we must consider connectivity. To this end, we must need special techniques to incorporate connectivity and error rates at the same time in the edge selection strategy.
4.2 Cost Functions Based on Error Feedback:

In our variations we modify the cost a single CNOT gate, or a single swap gate for both baselines. We do not modify how to determine which gates should be considered for heuristic cost in each baseline. To calculate heuristics cost for variations of baseline 1, we also consider the gates of current layer and the immediately next layer. However, now cost of each gate is different from other gate. In fact, cost of a gate most of the time the actual error rates of the real device. In case of baseline 2, we consider all of the gates of the immediately next groups of the considered group according the DAG of groups.

4.2.1 Fixed Costs:

The fixed cost function $g(x)$ is same for both baselines. At any state, already used swap gates are considered to calculate the fixed cost. We additionally incorporate readout error rates in the fixed cost function sometimes. The following is the description of these costs.

Swap Cost: Swap cost is static in the original baselines. To incorporate real time error feedback from the real device, we sum up the error rates of all basic gates needed to perform a swap operation. For a swap operation between two adjacent qubits, three CNOT gates are required. To apply a swap operation between two distant qubits, it is required to find the shortest path between two qubits. The weight of an edge is the cost of swap on the edge. All pair shortest path algorithm [9] is applied, to find the shortest path between every pair of qubits. This shortest path is the shortest path to move one qubit to another qubit. The path cost is the sum of cost of all swaps needed along the path. This path cost is the swap cost between two distant qubits.

Readout Cost: All the qubits in a quantum circuit must be measured to get results after execution of the circuit. This measurement operation also incurs error. This error rate of a qubit is considered as the readout cost of that qubit. The readout cost of a mapping is the sum of all readout error rates of all physical qubits involved in the mapping. The readout cost of a mapping can be added to swap cost to get more accurate fixed cost.

4.2.2 Heuristic Costs:

In the baselines heuristic costs are calculated by summing up static cost of all gates. However, in our variations we incorporate real time error rate of each gate to calculate the cost of corresponding gate. CNOT gate is the only multi-qubit gate. The cost of CNOT gate is defined as CNOT cost. In some of our variations we also considered the error rates single qubit gates. The cost of single qubit gate is defined as single qubit gate cost. In the following we provide elaborated description of these costs and the way they are calculated.

CNOT cost: In the case of baseline 1 CNOT gates of current layer and the next layer is considered. In the case of baseline 2 CNOT gates of the immediately next groups according to the circuit DAG are considered. Some CNOT gates among the considered CNOT gates can be applied directly according to the current mapping. Some CNOT gates can not be applied directly. If the mapped physical qubits of the logical qubits of a CNOT gate is adjacent in the device, then CNOT gate can be applied directly. In this case CNOT cost of the gate is the CNOT error rate of the edge where the gate is going to be applied. On the other hand, if the mapped physical qubits are not adjacent in the real device, then we can not apply the CNOT gate directly. To resolve this, we need to consider the shortest path between these two physical qubits. The shortest path is determined by considering edge cost as swap cost. Then we apply the CNOT gate at the position which will minimize the overall cost in this path. Therefore, this path cost is the cost of the CNOT gate i.e., the CNOT cost.

Single qubit gate cost: Each qubit in the real device has a single qubit gate error rate. If a single qubit gate is applied to a physical qubit of real device, the single qubit gate error rate of that qubit is the single qubit gate cost.

4.3 Edge Selection Strategies:

Every mapping algorithm needs to choose edges from the coupling map of real device to determine initial mapping and to determine necessary swaps to get a new mapping from a given mapping. Selection of proper edges reduces overall circuit size. Hence, the probability of success increases. The selection of an edge is done considering some criteria, such as connectivity of an edge, real time error parameters etc. In the following we describe three strategies: baseline strategy, error sensitive (ES) strategy, and connectivity aware error sensitive (CAES) strategy.

Baseline Strategy: In the baselines, all the edges are considered same. Therefore, edges are chosen according to the natural order. The natural order means the order in which the edges naturally are in. We say this as no edge ordering strategy.

Error Sensitive (ES) Strategy: In this approach each edge has a cost associated with it. The edges with smaller cost are chosen earlier. The cost of an edge is calculated using real-time error parameters associated with the edge. Let an edge $(a, b)$, where $a$ and $b$ are the associated physical qubits of that edge. Now cost$(a, b)$ is calculated as the sum of single qubit error rates and readout error rates of qubit $a$ and qubit $b$. The edge ordering according to this strategy is termed as simple edge ordering.

Connectivity Aware Error Sensitive (CAES) Strategy: Considering only the error parameters of an edge is not always good. An edge having lower (better) error rates can have worse connectivity. Connectivity of an edge is defined by the connectivity of the physical qubits associated with the
edge. The higher number of other qubits directly connected to a qubit, the more connectivity the qubit has. According to the connectivity graph (coupling map) of a device the degree of a node defines its connectivity. The higher the degree a node (qubit) has, the greater connectivity it has. If the qubits associated with an edge has higher connectivity then the edge also has higher connectivity. If an edge has higher connectivity, then choosing this edge earlier will reduce the number of additional swaps. Because additional swaps are needed due to having less connectivity. For example, if all the qubits are connected directly to every other qubits, then no additional swaps will be needed. Therefore, connectivity of an edge \((a, b)\) can be calculated as degree\((a) + \text{degree}(b)\), i.e., the sum of the degree of the nodes associated with the edge. For example, in figure 4 connectivity of node 8 is 3 and node 7 is 1. Connectivity of edge \((8, 7)\) is 4.

Considering only connectivity may reduce the probability of success of the program. Because an edge having higher connectivity may have higher (worse) error rates. Additionally an edge itself may have lower (better) error rates. However, the edges incident on the qubits of the edge can have higher (worse) error rates. Therefore, we should consider connectivity with real-time error feedback. To this end, we can consider weighted connectivity. We know that the coupling map is a weighted graph. Error rate of an edge is considered as the weight of the edge. Weighted connectivity of an edge is calculated as the sum of the weights of the edges incident on qubits associated with the edge. Now the question is which edge we should choose earlier. The edge with higher value of weighted connectivity can be chosen earlier. However, it has a fatal drawback. Since error rate is taken as weight, higher value of weighted connectivity can occur due to high error rates of the incident edges. As a result overall probability of success will be reduced. Even an edge having lower connectivity can have higher weighted connectivity if some of the connected edges have very high error rates. As a result performance will be worst. On the other hand, choosing an edge having lower value of weighted connectivity is also not desired. Because lower value of weighted connectivity occurs due to having actually lower connectivity i.e., the qubits associated with the edge have smaller number of connections with other qubits.

Consequently, we consider another way to incorporate connectivity and real-time error feedback simultaneously. To this end, we define a cost function for each edge. Cost of an edge \((a, b)\) is defined as,

\[
\text{cost}(a, b) = \sum_{i \in \text{all qubits except } a,b} \left(\text{dist}(a, i) + \text{dist}(b, i)\right)
\]

Here \(\text{dist}(a, b)\) is the shortest path distance between qubit \(a\) and qubit \(b\), where edge weight is the CNOT error rate. In this policy the edge with smaller value of cost is chosen earlier. The question is how the approach can incorporate connectivity and error feedback simultaneously. The qubit having higher connectivity must have smaller value of sum of the shortest path distances to every other qubits. Because the qubit is already connected to many other qubits directly due to its higher connectivity. As a result shortest path costs to these directly connected qubits are the cost of one edge only i.e., very small shortest path costs for all connected qubits. Therefore, sum of the shortest path costs is smaller. On the other hand, since, error rate is considered as the weight of an edge, shortest path minimizes the overall error along a path. As a result minimizing sum of shortest path costs minimizes overall error, resulting in higher probability of success (fidelity). Therefore, taking edge with smaller cost, according to this cost function, earlier will ensure higher connectivity and lower (better) error rates at the same time. As a result number of additional swaps will be decreased and fidelity will be increased. The edge ordering according to this strategy is termed as special edge ordering.

### 4.4 Variations of Error-Sensitive Mappings:

We consider 12 variations of baseline 1 and 9 variations of baseline 2 to incorporate real-time error feedback in order to get near optimal fidelity in the quantum circuits. We do not modify the original baseline algorithms in any of the variations. We also do not modify how different gates are considered to calculate cost functions, and the A* algorithm used by both baselines. We just focus on how the cost of each gate is calculated. In case edge selection, we just modify the order in which different edges are chosen at different stages of the algorithms. In fact, all cost functions in both baselines are static. On the other hand, in our variations we consider real-time error sensitive dynamic cost functions. And there are no orders in which different edges of the real device should be chosen in the baselines. However, we consider different orderings or strategies to choose edges based on real-time error feedback and connectivity of the device. Cost function has two parts: fixed costs and heuristics costs. We described
earlier in this paper different real-time error sensitive fixed cost functions and heuristic cost functions, and different edge selection strategies. In each of our variation we are going to consider these different cost functions and edge selection strategies. More elaborately, we have two different fixed costs such as swap cost and readout cost. In a variation we can consider swap cost only or readout cost only or both costs at the same time. If we consider both costs at the same time then two costs will be summed up to get total fixed cost. The approach is same in calculation of heuristic cost also. In case of edge selection strategy we consider a single strategy in a variation among the three different strategies. In case of baseline 2 we consider another parameter said as group ordering. Baseline 2 works by topologically sorting groups according to the DAG. Since there are multiple possible topological order, we have opportunities to prefer a specific topological order based on the some criteria. This approach of group selection is called group ordering. More elaborately, In case of determining topological order, sometimes it happens that two or more groups have same preference. That means any one of these groups can be considered earlier in the topological order. For example in the figure 1(d), we can consider $G_1$ or $G_2$ as the first in topological order. Again we can consider any one of $G_3$, $G_4$, and $G_5$ earlier in topological ordering. In the original baseline 2 there is no group ordering. Naturally one group is chosen earlier without considering any criterion. To this end, we adopt a criterion for group ordering in some of the variations of baseline 2. The criterion is: the group having larger number of CNOT gates will get more preference over others having smaller number of CNOT gates. All of our variations on baseline 1 is shown in table 1. And all of the variations on baseline 2 is shown in table 2.

5 Experimentation:

We performed our experimentation in two phases. At first we run all of our variations of both baselines on random benchmarks. Then the best mapping algorithm among both baselines is compared with the state of the art heuristic based mapping algorithm provided by [20] on representative benchmarks ( typical programs ) used by QC systems.

Setup: We performed our experiments in the quantum computer of IBM available online named as ‘ibmq_16_melbourne’ , which is situated at Melbourne, Australia. All codes to find different mappings are written in C++. However, all the scripts are run using python to perform the experimentation. We used IBM Quantum Experience API to access calibration data or real-time error feedback. All of our experimentation on random benchmarks are completed within one week from 10th october, 2018. However, the experimentation to compare with [20] are performed in May, 2020.

Since there is no open source implementation of the best edge based heuristic algorithm provided in [20], we have implemented it in C++ according to the description provided in [20].

Random Benchmarks: We use random benchmarks to evaluate various heuristics mapping in baselines. Total twelve benchmarks are used as in table 3. The circuit width ( number of used qubits ) varies from 4 to 7. The depth varies from 4,6 and 8 for circuit width 4,5,6. The depth varies from 5,6,7 for circuit width 7. Since the width is higher, so depth is not taken to 8. All the circuits are generated by sampling from cx, x, h gates. However, this sampling was done in a way, so that final output will have only one state i.e., no superposition output is considered here. In each layer, single qubit gates and CNOT gates were taken with equal probability. However, to create superposition, h gate is applied to all the used qubits and if h gate is applied in the first layer, h gates also applied in the last layer. All the qubits of a given width are used in each layer i.e., in each layer on each used qubit, either CNOT gate or single qubit gate is applied.

To get general result that can be implied for general quantum programs we have taken random circuits instead of some fixed quantum programs to evaluate various error sensitive heuristic mapping algorithms. For example, in some quantum programs the number of single qubit gates is smaller than that of CNOT gates. In that case, we cannot observe the effect of single qubit gates. However, in some quantum programs the number of CNOT gates is smaller than that of single qubit gates. To this end, to observe the effect of single qubit gates and multi-qubit qubit gates simultaneously, we sampled CNOT gates and single qubit gates with equal probability from uniform distribution. The benchmarks in [40] are very large in most of the cases to run into real devices. In fact, in IBM’s competition the benchmarks were randomly generated circuit [41].

Representative Benchmarks: Though we have evaluated our various error sensitive mappings on random benchmarks, traditional compilers are generally evaluated on typical workloads that are representative of some domain. Therefore, to compare our best heuristic mapping with the state of the art mapping [20], we choose typical quantum circuits those are kernels of many important quantum algorithms. Our 12 representative benchmarks were chosen from previous works on compilation and system evaluation [1], [18], [31]. In fact, most of these benchmarks were chosen to evaluate the optimal and heuristics approaches in [20]. Also many of these benchmarks are kernels of many important quantum algorithms [20]. The characteristics of these benchmarks are shown in table 4.

Metrics: The metric is probability success i.e., success rate or fidelity. We run each of variation on each benchmarks 8192 times in real device. The number of successful outputs are divided by 8192, to get the probability of success or fidelity. To evaluate various heuristics, for each variation, average success rate over all random the benchmarks is considered as the performance measure of the variation. The same approach is considered in the case of comparison with the state of the art. The same metric is adopted to evaluate the algorithms.
## Cost Functions

| Variation # | Fixed Costs | Heuristics Costs | Edge Selection Strategy | Comments |
|-------------|-------------|------------------|--------------------------|----------|
| 1.1         | baseline    | baseline         | baseline                 | Naive    |
| 1.2         | swap cost   | baseline         | baseline                 |          |
| 1.3         | baseline    | baseline         | ES                       |          |
| 1.4         | baseline    | baseline         | CAES                     |          |
| 1.5         | baseline    | CNOT cost        | baseline                 |          |
| 1.6         | swap cost + readout cost | CNOT cost | baseline                 |          |
| 1.7         | swap cost + readout cost | CNOT cost | ES                       |          |
| 1.8         | swap cost + readout cost | CNOT cost | CAES                     |          |
| 1.9         | swap cost   | CNOT cost + single qubit gate cost | baseline |          |
| 1.10        | swap cost + readout cost | CNOT cost + single qubit gate cost | ES       |          |
| 1.11        | swap cost + readout cost | CNOT cost + single qubit gate cost | baseline |          |
| 1.12        | swap cost + readout cost | CNOT cost + single qubit gate cost | CAES     | CAES     |

Table 1: Variations based on Baseline 1

| Variation # | Fixed Costs | Heuristics Costs | Edge Selection Strategy | group order |
|-------------|-------------|------------------|--------------------------|-------------|
| 2.1         | baseline    | baseline         | baseline                 | None        |
| 2.2         | swap cost   | baseline         | baseline                 | None        |
| 2.3         | swap cost   | CNOT cost        | baseline                 | None        |
| 2.4         | swap cost   | CNOT cost        | CAES                     | None        |
| 2.5         | swap cost + readout cost | CNOT cost | CAES                     | None        |
| 2.6         | swap cost + readout cost | CNOT cost | CAES                     | None        |
| 2.7         | swap cost + readout cost | CNOT cost + single qubit gate cost | CAES     | None        |
| 2.8         | swap cost + readout cost | CNOT cost | CAES                     | Used        |
| 2.9         | swap cost + readout cost | CNOT cost + single qubit gate cost | CAES     | Used        |

Table 2: Variations based on Baseline 2

| circuit # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|-----------|---|---|---|---|---|---|---|---|---|----|----|----|
| width     | 4 | 4 | 4 | 5 | 5 | 5 | 6 | 6 | 7 | 7  | 7  |    |
| depth     | 6 | 8 | 10| 6 | 8 | 10| 6 | 8 | 10 | 5  | 6  | 7  |

Table 3: Characteristics of Random Benchmarks

in [20, 32]. We do not consider mapping time. Because our goal is how to incorporate real-time error feedback and connectivity information effectively in order to get higher fidelity in outputs.

### 6 Results:

We have first evaluated various heuristics and then compared our best heuristic method with the state of the art heuristic method [20]. The following is discussion on the results in each case.

![Figure 5: Average probability of success over different approaches based on baseline 1.](image-url)
| Name | Width | CNOTs | Single Qubit Gates | CNOT graph |
|------|-------|-------|--------------------|-------------|
| Adder | 4     | 10    | 13                 | ![CNOT graph](image) |
| BV4  | 4     | 3     | 9                  | ![CNOT graph](image) |
| BV6  | 6     | 3     | 9                  | ![CNOT graph](image) |
| BV8  | 8     | 3     | 15                 | ![CNOT graph](image) |
| HS2  | 2     | 2     | 14                 | ![CNOT graph](image) |
| HS4  | 4     | 4     | 24                 | ![CNOT graph](image) |
| HS6  | 6     | 6     | 36                 | ![CNOT graph](image) |
| Or   | 3     | 6     | 11                 | ![CNOT graph](image) |
| Peres| 3     | 5     | 11                 | ![CNOT graph](image) |
| Toffoli | 3   | 6     | 12                 | ![CNOT graph](image) |
| Fredkin | 3   | 8     | 11                 | ![CNOT graph](image) |
| Shor | 4     | 9     | 4                  | ![CNOT graph](image) |

Table 4: Characteristics of Typical Workloads (Representative Benchmarks)

Figure 6: Average probability of success over different approaches based on baseline 2

### 6.1 Evaluation of Various Heuristics:

We run all the experiments in the real devices like IBM’s quantum experience. We run experiments to observe the effects of different real-time error parameters. Our results show that the incorporation of different error parameters improve fidelity in real device. However, without edge ordering or connectivity awareness fidelity improvement is substantially lower than that of with edge ordering. The best performance we get in both baselines whenever special edge ordering (CAES) is used with all other error parameters. In case of variations based on baseline1, we got an average 1.62x improvement over [40]. In the case of variations based on baseline2, we got an average 1.934x improvement over [41].

In these experimentation our main focus is not on algorithmic development. Our main target is to validate how different error rates and heuristics affect the fidelity or performance of quantum circuits experimentally. We get very promising results incorporating these heuristic techniques. This empirical result shows that, incorporating all of the error parameter should give higher fidelity results than leaving some thing like single qubit gate error rates. For example in 1.8 we considered everything except single qubit gates error rate. On the other hand, in 1.12 we considered single qubit error rate also. We get better performance in 1.12 after considering single qubit gate error rates. This shows that consideration of single qubit gate error rate matters. In comparison with 1.10 with 1.12 (CAES), we observe that the only difference is 1.10 uses only ES edge selection strategy and 1.12 uses CAES edge selection strategy. Results show that 1.12 performs 25% better than 1.10. Therefore, connectivity consideration is very important. Additionally, incorporation of every error rates such CNOT error rates, single qubit gates error rates, readout error rates are necessary to get better fidelity outputs. Moreover, we can observe the performance of 1.9 is much lower than 1.12. The only difference between these is, in 1.9 no edge ordering is considered, but in 1.12 special edge ordering is considered (CAES). This difference caused a substantial difference in performance. Therefore, considering connectivity aware strategy is of utmost importance. Because without considering connectivity performance may be very bad. The reason behind this is explained earlier. Now it is validated by experiments.

In case various mappings based on baseline 2. We get similar results as in baseline 1. Since connectivity aware error sensitive (CAES) performs always better, we use this in most of the variations of baseline 2 from 2.4 to 2.9. Therefore, the fidelity or success rate is similar in all of these variations. The variations in 2.7 and 2.9 are almost similar except in 2.9 group ordering is used. It can be observed that the probability of success or success rate is a little bit better in 2.9. Though the improvement in 2.9 due to use of group ordering is small, it indicates the group ordering is useful. In fact, group ordering is rarely used in these circuits as the circuits are very small. The effect of group ordering will be visible in large circuits. However, larger circuits can not be used at the current status of QC systems.

Finally, not only error rates but also connectivity information should be considered to get high fidelity outputs. All of these experiments and theoretical analysis show that connectivity aware error sensitive (CAES) mapping technique is utmost necessary when designing heuristic based mapping techniques.

### 6.2 Comparison with the state of the art:

CAES mapping algorithm that is our best heuristic approach (1.12) performed 1.7x (on average) better than the state of the art heuristic approach [20] on representative benchmarks.
These drawbacks are validated by our experiments also. In BV6, BV8, Shor etc. Since all of these benchmarks have larger connected to each other. We can see the effect in Adder, BV4, BV6, BV8 has lower connectivity. For example, benchmarks having only a single edge, or multiple disconnected edges are suitable for their approach. Therefore, it performs well in the benchmark HS2, HS4, HS6. If we see the CNOT graph in each of these benchmarks, we observe only a single edge is the maximum connectivity. This is the main reason because of that their algorithm performs worse in the workloads Adder, BV4, BV6, Shor, Or, etc. And therefore, our approach performs better than their approach in most of the benchmarks in a large margin. According to experimental results and also with theoretical analysis we can conclude that connectivity aware mapping algorithm or heuristics techniques is a must need to get higher fidelity in the real device. Without connectivity consideration performance would be worse.

Reliability metric: The reliability metric provided in [20] does not find the most reliable path as we explained earlier. This is validated by experiments. The mapping in [20] performs worse when the CNOT graph has more than 2 nodes connected to each other. We can see the effect in Adder, BV4, BV6, BV8, Shor etc. Since all of these benchmarks have larger CNOT graph, failing to find the most reliable path hinders performance. The simple reliability metric, we considered, is better than that of theirs. Because of that our best approach (CAES) gets better fidelity or reliability in the real device in most cases. Therefore, when designing real time error sensitive mapping algorithm, it is very important to consider perfect reliability metric. We consider just error rate as the weight cost. So path cost is taken as sum of error rates along the path. This is better than the approach provided in [20]. Moreover, in future we will design better reliability metric that will be more suitable.

Look-ahead scheduling: In the case of routing qubits, when a CNOT gate is applied between two distant qubits, consideration of only the shortest path between the two qubits should not suffice. After routing the mapping will be changed. The changed mapping must have effects on all of the subsequent gates. Without consideration of the effects on the subsequent gates, mapping will be very much unreliable. Because the mapping satisfying the current CNOT constraint, can be the worst mapping for the subsequent gates requiring many swaps. Therefore, it is utmost necessary to apply some look ahead scheme to measure the effect of the current mapping on the subsequent mapping. In our baselines, the look ahead scheme was considered. Therefore, we also considered look ahead scheme in all of variations in both baselines. However, the approach in [20] lacks of this consideration. Hence, get worse fidelity in the real device compared to our CAES mappings. Therefore, consideration of some look ahead scheme to estimate the effect on the subsequent gates is also very important.

From our theoretical analysis and experimental results it is clear that any real-time error sensitive mapping algorithm must have three properties to perform well: good reliability metric, connectivity awareness, and look ahead routing policy.

7 Discussions:

In this section we are going to discuss on the best heuristic approach (CAES) we devised. We are going to describe its optimality, generality, efficiency, limitations, and corresponding future works.

Optimality: The authors of [20] provided an optimal approach to the problem. So far this is the only optimal method. They also provided two heuristic methods. One of these heuristic methods performs better than the other. The best performing heuristic in [20] gets near optimal success rate [20]. This work in our paper is considered as the state of the art. Therefore, our CAES must be more near optimal in terms of fidelity or success rate. 

Generality: The generality of our developed heuristics techniques is already verified by application of our approach.
into two different baseline algorithms. We discussed earlier that we worked on two parts: cost functions and edge selection strategies. The cost functions or edge selection strategy we devised are independent of any mapping algorithm. They depend on the real-time error feedback or coupling map (connectivity graph) of the real device only. In fact, any real time error sensitive mapping algorithm must need to consider the real device error rates and must choose edges from the coupling map. Therefore, our techniques are not confined to some fixed mapping algorithm, rather applicable in any mapping algorithm in general.

In fact, every QC system must have qubits and connectivity between qubits. Only the implementation technology to realize qubits and connectivity between qubits may differ. Moreover, the existing QC systems like ion trap or superconducting suffer from errors [18]. Therefore, our developed techniques are applicable to any QC systems irrespective of the implementation technologies.

**Efficiency:** We discussed earlier that we do not change the computation technique of different heuristics functions and cost functions, rather we just consider real time error sensitive dynamic cost value of different parameters. Consequently, compilation time is not affected. In the case of different edge selection strategies, we need to order the edges of connectivity graph of real device. More specifically, we need to sort the edges of the real device according to some cost function associated with each edge. Since the sorting task is solely dependent on the device’s error rates and connectivity graph, it can be done in advance before compilation. In the course of compilation, the predetermined order of the edges will be used. Therefore, our heuristics techniques do not affect compilation time. As a result, these techniques are applicable very efficiently in general. Moreover, calculation of heuristics also could be done in polynomial time. For example we use Floyd Warshal’s all pair shortest path algorithm [9] with complexity $O(n^3)$. Calculation of our best heuristics technique CAES involves distances for all qubits for each edge. The complexity of this task is also $O(n^3)$. Sorting of edges could be done in $O(n^2 \log(n))$ time. Overall complexity of prior computation is $O(n^3)$ which is less than the actual mapping time. Calibration data from the real device can be collected at any moment using IBM Quantum Experience API. Therefore, there is no memory overhead.

**Limitations:** In this work, we are mainly concerned with developing heuristic techniques in which we can incorporate real time error feedback and connectivity information of the device simultaneously. To this end, we worked on two baselines. In fact, we did not focus on algorithmic development. As a result, all the algorithmic limitations that can be applicable in the baselines are not diminished. For example our baseline 1 [40] some times fails to find mapping within reasonable time. Because the A* algorithm is not admissible. Hence, it incurs sometimes exponential complexity. For small circuits with limited depth it works fine. Therefore, scalability can not ensured using these baselines. To ensure scalability we need to incorporate our heuristics techniques in some scalable baselines. Another thing is that in this work we did not consider gate duration of real device and also cross-talk between physical qubits.

**Future Works:** We have a couple of targets in future. First of all we will devise scalable mapping algorithms which incorporates our developed techniques to get high fidelity mapping. In that case, we may work on some scalable baseline or we may need to design our own algorithm. The optimal approach provided in [20] is not scalable beyond 32 qubits. Therefore, we can improve that optimal algorithm using our developed techniques as well as other techniques also. For example, SMT solvers are used in that optimal approach. We may consider application of more elegant solvers with better scalability. Finally, we have a plan to work on more accurate error model. For example, We will consider cross-talk between qubits, gate duration etc. Also we will work on more accurate and perfect reliability metric. Another important thing is feedback from circuit. In our approaches we mostly taken feedback from real device. However, it is possible to analyze the circuit and perform mapping task according to the analyzed data. Therefore, in calculation of our best heuristic method (CAES) we may incorporate feedback from circuit.

**8 Conclusions:**

Since the computing power of classical computers is at the verge of its limit, shifting to new more powerful computing systems is utmost necessary. QC systems are greatly promising for this paradigm shift. It is becoming closer to the supremacy over the classical computing systems day by day. Therefore, it is high time for developing compilers and related system software for this newly emerging computing systems. In this paper we worked on the most crucial part of quantum compiler, that is quantum circuit mapper. We developed highly elegant techniques to improve fidelity of quantum circuits. Our developed best method performs substantially better than the state of the art. We also discovered criteria for quantum circuit mapper to perform well. In future we have plans to work on scalable high fidelity mapping algorithms for quantum compiler, in order to move forward the highly promising newly emerging QC technologies.

**References**

[1] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):818–830, 2013.

[2] Amlan Chakrabarti, Susmita Sur-Kolay, and Ayan Chaudhury. Linear nearest neighbor synthesis of re-
versatile circuits by graph partitioning. *arXiv preprint arXiv:1112.0564*, 2011.

[3] Byung-Soo Choi and Rodney Van Meter. On the effect of quantum interaction distance on quantum addition circuits. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 7(3):11, 2011.

[4] Simon J Devitt. Programming quantum computers using 3-d puzzles, coffee cups, and doughnuts. *arXiv preprint arXiv:1609.06628*, 2016.

[5] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[6] Mohammad Javad Dousti and Massoud Pedram. Minimizing the latency of quantum circuits during mapping to the ion-trap circuit fabric. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 840–843. IEEE, 2012.

[7] Richard P Feynman. *Feynman lectures on computation*. CRC Press, 2018.

[8] Will Finigan, Michael Cubbeddu, Thomas Lively, Johannes Fickl, and Prineha Narang. Qubit allocation for noisy intermediate-scale quantum computers. *arXiv preprint arXiv:1810.08291*, 2018.

[9] Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[10] Mrityunjay Ghosh, Amlan Chakrabarti, and Niraj K Jha. Automated quantum circuit synthesis and cost estimation for the binary welded tree oracle. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(4):51, 2017.

[11] Gian Giacomo Guerreschi and Jongsoo Park. Two-step approach to scheduling quantum circuits. *Quantum Science and Technology*, 3(4):045003, 2018.

[12] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[13] IBM. Ibm quantum experience. [Online; accessed 10-May-2020].

[14] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242, 2017.

[15] Abhoy Kole, Kamalika Datta, and Indranil Sengupta. A new heuristic for $n$-dimensional nearest neighbor realization of a quantum circuit. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):182–192, 2018.

[16] Chia-Chun Lin, Amlan Chakrabarti, and Niraj K Jha. Fuqs: Fault-tolerant quantum logic synthesis. *IEEE Transactions on very large scale integration (VLSI) systems*, 22(6):1350–1363, 2013.

[17] Chia-Chun Lin, Susmita Sur-Kolay, and Niraj K Jha. Paqcs: Physical design-aware fault-tolerant quantum circuit synthesis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(7):1221–1234, 2014.

[18] Norbert M Linke, Dmitri Maslov, Martin Roetteler, Shantanu Debnath, Caroline Figgatt, Kevin A Landsman, Kenneth Wright, and Christopher Monroe. Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences*, 114(13):3305–3310, 2017.

[19] Andrea Morello and David Reilly. What would you do with 1000 qubits. *Quantum Science and Technology*, 3(3):030201, 2018.

[20] Prakash Murali, Jonathan M. Baker, Ali Javadi-Abhari, Frederic T. Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS ’19, pages 1015–1029, New York, NY, USA, 2019. ACM.

[21] Massoud Pedram and Alireza Shafaei. Layout optimization for quantum circuits with linear nearest neighbor architectures. *IEEE Circuits and Systems Magazine*, 16(2):62–74, 2016.

[22] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O’Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5:4213, 2014.

[23] Paul Pham and Krysta M Svore. A 2d nearest-neighbor quantum architecture for factoring in polylogarithmic depth. *Quantum Information & Computation*, 13(11-12):937–962, 2013.

[24] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, Aug 2018.

[25] Miha Rot. Quantum machine learning.
[26] Daniel Ruffinelli and Benjamin Baran. A multiobjective approach to linear nearest neighbor optimization for 2d quantum circuits. In 2016 XLII Latin American Computing Conference (CLEI), pages 1–8. IEEE, 2016.

[27] Mehdi Saeedi, Robert Wille, and Rolf Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures. *Quantum Information Processing*, 10(3):355–377, 2011.

[28] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures. In *Proceedings of the 50th Annual Design Automation Conference*, page 41. ACM, 2013.

[29] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[30] Marcos Yukio Siraichi, Vinicius Fernandes dos Santos, Sylvain Collange, and Fernando Magno Quintão Pereira. Qubit Allocation. In *CGO 2018 - International Symposium on Code Generation and Optimization*, pages 1–12, Vienna, Austria, February 2018.

[31] Mathias Soeken, Thomas Haener, and Martin Roetteler. Programming quantum computers using design automation. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 137–146. IEEE, 2018.

[32] Swamit S. Tannu and Moinuddin K. Qureshi. Not all qubits are created equal: A case for variability-aware policies for nisq-era quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS ’19, pages 987–999, New York, NY, USA, 2019. ACM.

[33] Christophe Vuillot. Is error detection helpful on ibm 5q chips? *arXiv preprint arXiv:1705.08957*, 2017.

[34] Mark Whitney, Nemanja Isailovic, Yatish Patel, and John Kubiatowicz. Automated generation of layout and control for quantum circuits. In *Proceedings of the 4th international conference on Computing frontiers*, pages 83–94. ACM, 2007.

[35] Wikipedia contributors. Copenhagen interpretation — Wikipedia, the free encyclopedia, 2019. [Online; accessed 15-August-2019].

[36] Wikipedia contributors. No-cloning theorem — Wikipedia, the free encyclopedia, 2019. [Online; accessed 15-August-2019].

[37] Wikipedia contributors. Quantum entanglement — Wikipedia, the free encyclopedia, 2019. [Online; accessed 15-August-2019].

[38] Wikipedia contributors. Quantum logic gate — Wikipedia, the free encyclopedia, 2019. [Online; accessed 15-August-2019].

[39] Xin Zhang, Hong Xiang, Tao Xiang, Li Fu, and Jun Sang. An efficient quantum circuits optimizing scheme compared with qiskit. *arXiv preprint arXiv:1807.01703*, 2018.

[40] Alwin Zulehner, Alexandru Paler, and Robert Wille. An efficient methodology for mapping quantum circuits to the ibm qx architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[41] Alwin Zulehner and Robert Wille. Compiling su (4) quantum circuits to ibm qx architectures. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, pages 185–190. ACM, 2019.