An Analysis of the Importance of Variability-Aware Policies for NISQ-Era Quantum Computer

Shuohao Ping
sp1831@scarletmail.rutgers.edu

Department of Computer Science, Rutgers University

ABSTRACT

For papers focused on qubit mapping problem, most of them assume the connections between physical qubits in quantum hardware are the same. In other words, the error rate of a two-qubit operation on any two connected physical qubits is the same, like [1] and [2]. So, we can ensure the fidelity of quantum programs by minimizing the number of gates. However, [3] had shown that the connections between physical qubits also have different strengths, which means the error rate of two-qubit gates can be different for different pairs of connected physical qubits in quantum hardware. This implies that we want to map our logical qubits to physical qubits with strong connections to reduce errors caused by gates. Moreover, [4] had shown that the measurement error also differs for different qubits. The average error rate for measurement is around 4.7%, with a range from 0.85% to 22.2%. We need to take into account these errors in order to ensure the fidelity of quantum programs. Hence, the variability-aware policies in this paper will include gate error and measurement error, which is different from [3] that proposed the original variability-aware policies.

In this paper, I will discuss the way to modify SABRE, a SWAP-based qubit mapping algorithm for NISQ-era quantum computers, to make it variability-aware, which means the algorithm will consider different strengths of connections between physical qubits and measurement errors for each individual qubit. I will also discuss the impact of variability-aware policies on NISQ-era quantum computers.

1. INTRODUCTION

Quantum computing has received tremendous attention in the past few years. This led to the rapid development of quantum hardware and an exponential growth in the number of available qubits in quantum computers. By the summer of 2022, IBM had developed a 433-qubits quantum computer [5]. But these physical qubits are not fully connected in the quantum computer, and quantum programs assume every pair of qubits are connected. This led to the qubit mapping problem, which requires us to modify the original circuit of the quantum program in order to satisfy the constraints caused by limited connectivity between physical qubits. This problem is proved to be an NP-complete problem [6], which means the time needed to find an optimal mapping from logical qubits to physical qubits grows exponentially as the number of qubits in the quantum program grows. Hence, many prior works focused on developing technics to obtain a mapping that either reduces the number of additional gates or maximizes parallelism to reduce runtime within polynomial time. In those papers, connections between physical qubits in quantum computers are represented by an unweighted directed graph like Fig. 1.

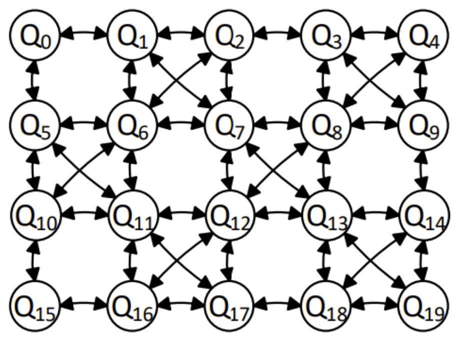


Fig. 1 IBM Q20 Tokyo Information [7]

In the above graph, nodes are denoted as Q_0 to Q_{19}, which represented physical qubits in IBM Tokyo quantum computers. The edges represent connections between physical qubits. In order to perform a two-qubit operation, such as a CNOT gate, on two qubits, those qubits have to be connected by an edge in the graph. For example, based on the above graph, we can apply a CNOT gate on qubits Q_0 and Q_1, but not on Q_0 and Q_6 because Q_0 and Q_6 are not connected by an edge in Fig. 1. Note that there are multiple-qubit operations that involve more than two qubits, like the Toffoli gate that uses three qubits, but those operations can be decomposed into a sequence of single-qubit and two-qubit operations. So, it is safe to consider only two-qubit operations in the qubit mapping problem.

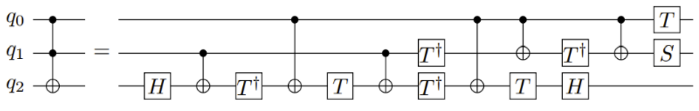


Fig. 2 Quantum Circuit of Toffoli Gate [1]

In fact, any quantum operations can be decomposed into

CNOT, H, S, and T gates. This set is called a Clifford set, and the gates in this set are universal quantum gates. For quantum programs, we assume qubits are fully connected, so we can apply any quantum operations on any qubits without concerning the limitations caused by hardware. Thus, it's common to encounter a situation of applying a two-qubit operation on two qubits without physical connections. In this case, a common technique is to apply SWAP gates to bring two qubits close to each other such that there is a physical connection between the two qubits after applying SWAPs. A SWAP gate is composed of three CNOT gates, as shown in the following graph.

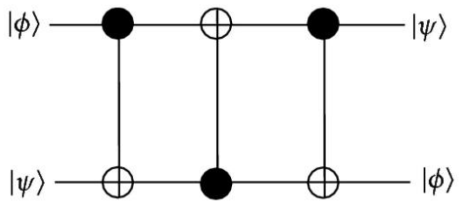


Fig. 3 Quantum Circuit of SWAP Gate

As the above graph shows, a SWAP gate can exchange the states of two qubits. Since information is encoded as states on qubits, the information on two qubits is exchanged after the SWAP gate. Now, let's reconsider the example of applying a two-qubit operation on Q_0 and Q_6. We can apply a SWAP gate on Q_0 and Q_1 and apply the two-qubit operation on Q_1 and Q_6 instead. In this case, we change the qubit mapping because the information originally stored on Q_0 is now moved to Q_1. Alternatively, we can also apply a SWAP gate between Q_0 and Q_5 and apply the two-qubit operation on Q_5 and Q_6. Since this unweighted graph assumes all connections have the same error rate, the two swappings are equivalent. However, connections between different physical qubits can have different strengths, or error rates, as shown by the following graph.

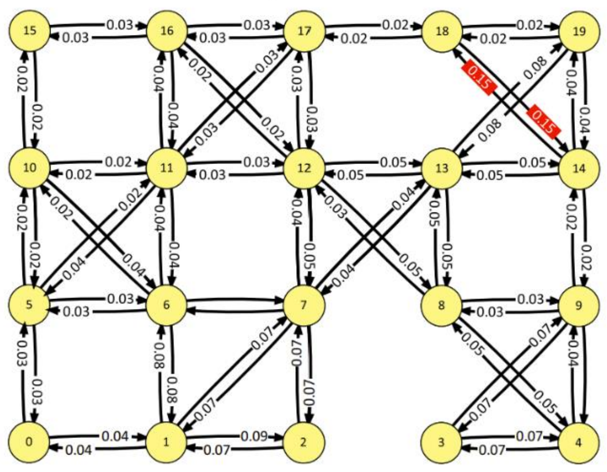
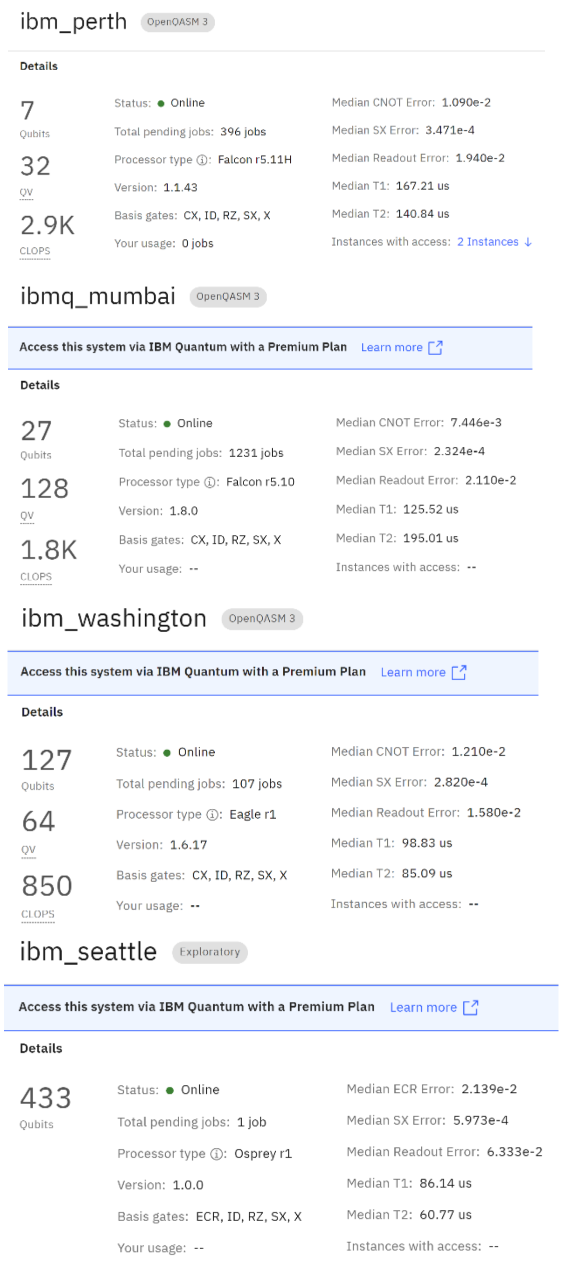


Fig. 4 Layout of IBM's 20 Qubit Machine with Label on Each Edge Representing the Probability of Failure on That Link [3]

In Fig.4, it's clear that the connections between physical qubits varied. The best connections have an average error rate of 2%, while the worst connections have an average error rate of 15%, which is more than 7 times the average error rate of the best connections. So, we want to avoid using the link with the highest error rate, even if that means using more SWAPs. Note that most quantum computers only support CNOT for two-qubit operations, as shown by the graphs below, and all other two-qubit operations must be decomposed into a sequence of universal gates before execution.



ibm_perth OpenQASM 3

Details

7 Qubits | 32 QV | 2.9K CLOPS

Status: ● Online
Total pending jobs: 396 jobs
Processor type ⓘ: Falcon r5.11H
Version: 1.1.43
Basis gates: CX, ID, RZ, SX, X
Your usage: 0 jobs

Median CNOT Error: 1.090e-2
Median SX Error: 3.471e-4
Median Readout Error: 1.940e-2
Median T1: 167.21 us
Median T2: 140.84 us
Instances with access: 2 Instances ↓

ibmq_mumbai OpenQASM 3

Access this system via IBM Quantum with a Premium Plan Learn more

Details

27 Qubits | 128 QV | 1.8K CLOPS

Status: ● Online
Total pending jobs: 1231 jobs
Processor type ⓘ: Falcon r5.10
Version: 1.8.0
Basis gates: CX, ID, RZ, SX, X
Your usage: --

Median CNOT Error: 7.446e-3
Median SX Error: 2.324e-4
Median Readout Error: 2.110e-2
Median T1: 125.52 us
Median T2: 195.01 us
Instances with access: --

ibm_washington OpenQASM 3

Access this system via IBM Quantum with a Premium Plan Learn more

Details

127 Qubits | 64 QV | 850 CLOPS

Status: ● Online
Total pending jobs: 107 jobs
Processor type ⓘ: Eagle r1
Version: 1.6.17
Basis gates: CX, ID, RZ, SX, X
Your usage: --

Median CNOT Error: 1.210e-2
Median SX Error: 2.820e-4
Median Readout Error: 1.580e-2
Median T1: 98.83 us
Median T2: 85.09 us
Instances with access: --

ibm_seattle Exploratory

Access this system via IBM Quantum with a Premium Plan Learn more

Details

433 Qubits

Status: ● Online
Total pending jobs: 1 job
Processor type ⓘ: Osprey r1
Version: 1.0.0
Basis gates: ECR, ID, RZ, SX, X
Your usage: --

Median ECR Error: 2.139e-2
Median SX Error: 5.973e-4
Median Readout Error: 6.333e-2
Median T1: 86.14 us
Median T2: 60.77 us
Instances with access: --

Fig. 5 Some of IBM's quantum computers. Note That CNOT, or CX, is the Only Two-qubit Gate in the Basis Gates. For ibm_seattle, the ECR Gate is the Two-qubit Gate in the Basis Gates. The ECR Gate is Equivalent to the CNOT Gate Up to Single-qubit pre-rotations.

This means the error rate of applying a SWAP gate on two qubits is not simply the error rate of the link since a SWAP gate must be decomposed into three CNOTS to execute on quantum computers. Assume the error rate for a particular link between two physical qubits is c where $0 < c < 1$. Then, the error rate of a SWAP gate on those two qubits is $1 - (1 - c)^3$ assuming errors on each CNOT gate are independent events. The reason for using this function is that the error won't occur only when all three CNOTs are executed without error. The following graph shows the error rate for a SWAP operation for IBM's 20 qubit machine using information from Fig. 4.

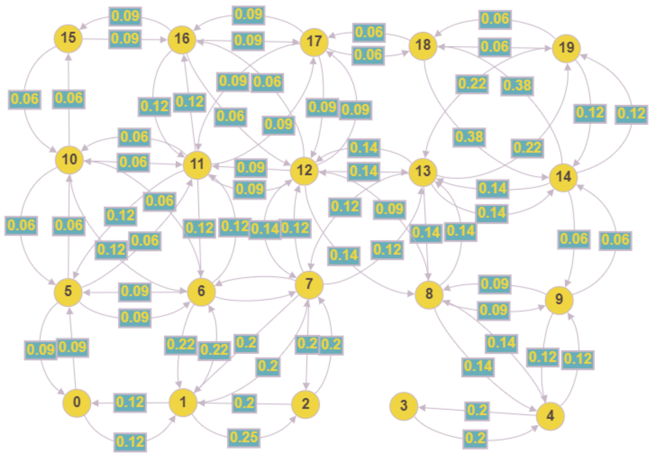


Fig. 6 Average Error Rate of SWAP for IBM's 20 Qubit machine Calculated using function $f(x) = 1 - (1 - x)^3$. The Labels for Edges between Node 6 and 7 Are Missing Due to Missing Labels in Fig. 4.

In Fig. 6, we saw that the connection between qubits 14 and 18 is so bad that a direct SWAP between 14 and 18 will have a larger error rate than applying two SWAPS on qubits 14 and 19 first and then 19 and 18 because the average error rate for two SWAPS can be calculated as $1 - (1 - 0.12)(1 - 0.06) = 0.1728 < 0.38$ assuming the failures on any physical link are independent events. Hence, if we want to SWAP qubits 14 and 18, we should perform two SWAPS instead of one to ensure the fidelity of the quantum program. This is counter-intuitive and not supported by most qubit mapping algorithms.

Lastly, we will need to consider the measurement error. In [4], the authors demonstrated that measurement errors for different physical qubits are different, as shown in the following graph.

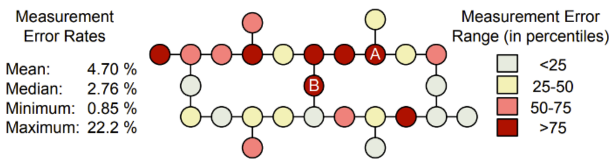


Fig. 7 Spatial Variation in Measurement Error Rates of Qubits on IBMQ-Toronto [4]

From the above graph, we can see that some of the qubits have a very high measurement error rate with a maximum of 22%. This would imply that we don't want to map a logical qubit that will be measured in the quantum program to a physical qubit that has a high measurement error. But we can still map a logical qubit that won't be measured during the entire program to the physical qubits with high measurement error. Those physical qubits can still be used for computation purposes, and most quantum algorithms don't need to measure all qubits by the end of the program. Furthermore, suppose a logical qubit is mapped to a physical qubit with high measurement error, like qubit B in Fig. 7. In that case, we can use SWAPs to swap the state of that qubit to a physical qubit with much lower measurement error and perform a measurement afterward. In the example of qubit B in Fig. 7, we can apply a SWAP gate between B and the physical qubit just below it, which has a much lower measurement error, and we will measure the lower qubit instead. This is enabled by the fact that we can perform measurements in the middle of a circuit execution [8], which means physical qubits with low measurement error can be reused for multiple measurements for different logical qubits by repeating the procedure of measuring physical qubits with low measurement error first and then using SWAPs to move the states of other logical qubits that need to be measured to that physical qubit and repeating the first step until all required measurements are done.

2. EVALUATION METHODOLOGY

2.1 Fidelity

For most of the prior works, fidelity is ensured by minimizing the number of additional SWAPs in the post-process circuit. Nevertheless, as I showed in the previous example, the least number of SWAPs doesn't guarantee the best fidelity for quantum programs. Sometimes, more SWAPs on reliable physical links are preferred compared to fewer SWAPs on unreliable physical links. Hence, I define fidelity as the probability of obtaining desired measurement results. This definition will take into account both gate and measurement errors. For simplicity, we will only consider these two types of error in our analysis, and we assume these errors happen independently. Also, we will ignore errors caused by bit flip and phase flip since those errors can be corrected by quantum error correction while gate and measurement errors can't. The goal of our algorithm should minimize the probability of obtaining incorrect results caused by gate and measurement error given a quantum program and a configuration of physical qubits in the quantum computer, including the average gate error rate for all connections between physical qubits and the average measurement error rate for each qubit. We will evaluate this criterion quantitatively.

2.2 Runtime

Another important evaluation criterion is the time complexity of our algorithm. We want our algorithm to have a polynomial time complexity even though we are not guaranteed to obtain the optimal result with polynomial time since this is an NP-complete problem. We will evaluate this criterion quantitatively.

3. VARIABILITY-AWARE POLICIES ON NISQ-ERA QUANTUM COMPUTER

3.1 Example algorithm

In this part, I will discuss how to modify SABRE's SWAP-based heuristic search algorithm proposed by [1] to make it variability-aware. There are four things we need to modify: distance matrix computing, initial mapping generation, new heuristic search algorithm, and lastly, I will add an algorithm to check for potential SWAPs for reducing measurement error. The inputs of this algorithm are a chip coupling graph with weights representing the error rate for each link, a quantum circuit, and a list of measurement errors for each physical qubit.

3.1.1 Computing Distance Matrix

Since we want to minimize the probability of measuring the incorrect state, it's natural to use distance in the graph to represent the probability of obtaining the incorrect state. Before we start to calculate the distance matrix, we need to make some changes to the input graph of physical qubits with gate error rates. Since the weight for each edge represents the error rate of a single two-qubit operation, and SWAP needs at least three two-qubit operations, we need to update the weight of each edge using function $\boldsymbol{f(x) = 1-(1-x)^3}$ where $\boldsymbol{x}$ is the original weight and $\boldsymbol{f(x)}$ is the new weight. This transformation ensures weight for each edge represents the error rate for a SWAP gate. The distance in this graph is the SWAP distance. Fig. 6 is an example of this transformation. We will also keep a copy of the input graph with the original weights, which will be used by other algorithms. Then, we can use Dijkstra's algorithm to find the shortest path between all physical qubits in the new weighted graph. This is the approach proposed by [3]. However, the weight on each edge represents a probability, and the summation of probabilities is not a probability. For example, if we sum up 11 edges with a weight 0.1, we will obtain a distance of 1.1, which is not a probability since probability can't be greater than 1. The correct result for the distance should be $\boldsymbol{1-(1-0.1)^{11} \approx 0.31}$, the probability of measuring incorrect results. But one can still argue that even though the distance is no longer representing a probability, a higher distance may still represent a higher error rate. This is also incorrect and can be shown by the following example.

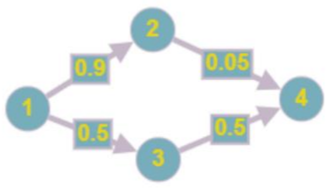


Fig. 8 An Example of Possible Layout for a Quantum Computer with Labels Representing Error Rate for Each Connection

Now, we want to find a path between node 1 and node 4 in Fig. 8 that has the least error rate. If we use summation, the shortest path will be $\boldsymbol{1 \rightarrow 2 \rightarrow 4}$ with a value of 0.95<0.5+0.5=1. However, if we calculate the error rate for the path $\boldsymbol{1 \rightarrow 2 \rightarrow 4}$, it is $\boldsymbol{1-(1-0.9)(1-0.05) = 0.905}$, and the error rate for the path $\boldsymbol{1 \rightarrow 3 \rightarrow 4}$ is $\boldsymbol{1-(1-0.5)(1-0.5) = 0.75}$, which is the correct shortest path. Therefore, it's necessary to redefine the distance function in Dijkstra's algorithm.

Let $\boldsymbol{X, Y}$ be two connected nodes in graph $\boldsymbol{G}$ with weight $\boldsymbol{0 \leq p \leq 1}$. Let $\boldsymbol{D(X,a), D(Y,a)}$ denote the distance between source $\boldsymbol{a}$ and X, Y. The relation is shown in the following graph.

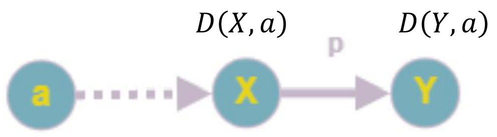


Fig. 9

Then, $\boldsymbol{D(Y,a) = 1-\big(1-D(X,a)\big)(1-p)}$ and $\boldsymbol{D(a,a) = 0}$. Under this definition, $\boldsymbol{D(Y,a)}$ represents the error rate of applying a sequence of two-qubit along some path from $\boldsymbol{a}$ to $\boldsymbol{Y}$ when $\boldsymbol{Y \neq a}$, and for any arbitrary path in the graph, the distance should not decrease as we move along the path. This can be proved.

$\boldsymbol{Proof\ by\ induction}$:

$Let\ G\ be\ a\ weighted\ graph\ such\ that\ for\ all\ weight\ q,\ \ 0 \leq q \leq 1.$

$Let\ k = \{s, p_1, p_2, \dots, p_n\}$ be an arbitrary path in G starting with node s.

$Let\ \ w(a,b)\ denote\ weight\ for\ edge\ connected\ node\ a\ and\ b.$

Base case:

$D(s,p_1) = \boldsymbol{1-\big(1-D}(s,s)\big)\big(\boldsymbol{1}-w(s,p_1)\big) = 1-(1-0)\big(1-w(s,p_1)\big) = w(s,p_1) \geq D(s,s) = 0.$

$Prob(link\ failed) = w(s,p_1) = D(s,p_1)$

Induction step:

Assume $D(s,p_i) = Prob(at\ least\ one\ link\ between\ s\ and\ P_i\ failed)$

Then, $D(s,p_{i+1}) = \boldsymbol{1-\big(1-D}(s,p_i)\big)\big(\boldsymbol{1}-w(s,p_{i+1})\big)$

$= 1 - Prob(no\ link\ between\ s\ and\ P_{i+1}\ failed)$

$= Prob(at\ least\ one\ link\ between\ s\ and\ P_{i+1}\ failed)$

Since $0 \leq w(s,p_{i+1}) \leq 1$, then $\big(\boldsymbol{1-D}(s,p_i)\big)\big(\boldsymbol{1}-w(s,p_{i+1})\big) \leq \big(\boldsymbol{1-D}(s,p_i)\big)$.

Hence, $-\big(\boldsymbol{1-D}(s,p_i)\big)\big(\boldsymbol{1}-w(s,p_{i+1})\big) \geq -\big(\boldsymbol{1-D}(s,p_i)\big)$.

Thus, $1-\big(\boldsymbol{1-D}(s,p_i)\big)\big(\boldsymbol{1}-w(s,p_{i+1})\big) \geq 1-\big(\boldsymbol{1-D}(s,p_i)\big)$.

Therefore, $\ D(s,p_{i+1}) \geq \boldsymbol{D}(s,p_i)$.

Algorithm 1: Modified Dijkstra's algorithm

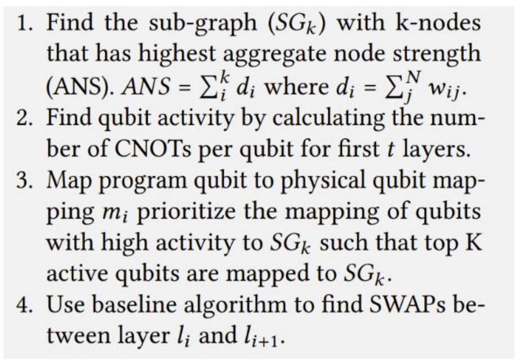
```
function Modified_Dijkstra(Graph, source):
    set Q to be a priority queue
    set S={}
    set dist[] to be a dictionary
    set prev[] to be a dictionary
    for each vertex v in graph.Vertices:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
```

```
dist[source] ← 0
add source to Q
add source to S
while Q is not empty:
    u ← vertex in Q with min dist[u]
    remove u from Q
        for each neighbor v of u still in Q:
            if v not in S then:
                add v to S
                temp ← 1-(1-dist[u]) *(1- Graph.Edges(u, v))
                if temp < dist[v]:
                    dist[v] ← temp
                    prev[v] ← u
return dist[], prev[]
```

The input graph is the graph with the SWAP distance since the distance matrix is used for finding SWAPs. A special case for Modified Dijkstra's algorithm is when all edges in the input graph have the same weight, which implies all connections have the same strength. In that case, the shortest path will have the least number of SWAPs, which is consistent with the assumption made by most prior works that minimizing the number of SWAPs increases fidelity.

3.1.2 Initial Mapping

In the original SABRE's SWAP-based heuristic search algorithm [1], the initial mapping is randomly generated. However, the author mentioned that initial mapping has a significant impact on the result of the final mapping. Hence, we can use the Variation-aware qubit allocation algorithm proposed by [3] to generate an initial mapping such that it prioritizes physical qubits with the strongest connectivity to the most used logical qubits in the given quantum programs. To obtain such allocation, we will modify the weight of each edge in the original input graph using function $g(x) = 1 - x$ so that the new weights represent the probability of success of a two-qubit operation. Then, the graph with updated weights and the circuit for the given quantum program will be used as the inputs for the Variation-aware qubit allocation algorithm shown below.



1. Find the sub-graph (SG_k) with k-nodes that has highest aggregate node strength (ANS). $ANS = \sum_i^k d_i$ where $d_i = \sum_j^N w_{ij}$.
2. Find qubit activity by calculating the number of CNOTs per qubit for first t layers.
3. Map program qubit to physical qubit mapping m_i prioritize the mapping of qubits with high activity to SG_k such that top K active qubits are mapped to SG_k.
4. Use baseline algorithm to find SWAPs between layer l_i and l_{i+1}.

Fig. 9 Variation-aware Qubit Allocation (VQA) Algorithm [3]

After we obtain the result from VQA, we will use it as the initial mapping for the SABRE search algorithm.

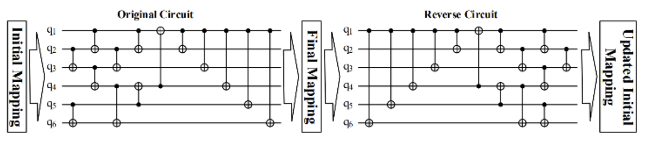
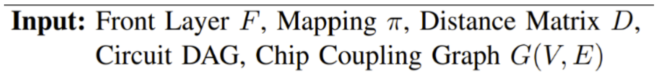


Fig. 10 SABRE Will Update the Initial Mapping Using Reverse Traversal Technique

3.1.3 Modified Heuristic Search Algorithm

In this part, I will modify the heuristic cost function in order to avoid mapping a logical qubit that needs to be measured to a physical qubit with high measurement error. This can be done by introducing an additional term in the cost function. But before we can introduce the new term, we need some additional inputs for the heuristic search algorithm.



Input: Front Layer F, Mapping π, Distance Matrix D, Circuit DAG, Chip Coupling Graph $G(V, E)$

Fig. 11 Inputs for Original Heuristic Search Algorithm

As shown in Fig. 11, the heuristic search algorithm needs a circuit DAG as its input. The original circuit DAG only contains information about dependencies of two-qubit gates. Now, I will extend circuit DAG such that it contains information about dependencies of both two-qubit gates and measurement. Then, we are ready to define the new term.

TABLE I: Definition and Notations used in the heuristic cost function.

Notation	Definition
$q\{1,2,\ldots n\}$	Logical qubits in quantum circuit
$Q\{1,2,\ldots n\}$	Physical qubits on quantum device
$D[][]$	Distance matrix for physical qubits. $D[i][j]$ is the distance between Q_i, Q_j
$\pi()$	A mapping from $q\{1,2,\ldots n\}$ *to* $Q\{1,2,\ldots n\}$
$\pi^{-1}()$	A mapping from $Q\{1,2,\ldots n\}$ *to* $q\{1,2,\ldots n\}$
F	Front layer
E	Extended set
$M[]$	Measurement error of physical qubits. $M[i]$ is the measurement error of Q_i

Since we still want to use physical qubits with high measurement error for computation, we will only consider the effect of measurement error when there is a measurement in E. Also, logical qubits that won't be measured during the entire program shouldn't be affected by the measurement error of physical qubits they map to. In fact, those logical qubits should be prioritized to be mapped to physical qubits with high measurement errors. Hence, based on these criteria, the new term is defined as the following:

$H_{measurement} = c * \sum_{measurement \in E}\big((measurement.q_i == SWAP.q_1) + (measurement.q_i == SWAP.q_2)\big) * M[\pi(measurement.q_i)]$

In the above function, c is a constant used for controlling the influence of measurement error on choosing SWAPs. Large c means measurement error will significantly influence the choice of SWAP. Normally, c should be small because errors caused by operations is often higher than measurement error due to a large number of operations. In addition, $(measurement.q_i == SWAP.q_1)$ and $(measurement.q_i == SWAP.q_2)$ are two Boolean functions that return either 0 or 1. If none of the two logical qubits used by SWAP will be measured in the near term (measurements occur in E), then $H_{measurement} = 0$, which implies that the cost of this SWAP is independent of the measurement error. On the other hand, if both logical qubits used by SWAP will be measured in the near term, then $H_{measurement} = c * (M[\pi(SWAP.q_1)] + M[\pi(SWAP.q_1)])$, which implies that the cost of this SWAP is strongly related to the measurement errors of the physical qubits used by this SWAP. Since we prefer SWAP with low heuristic cost, this will make logical qubits that are going to be measured be mapped to physical qubits with less measurement error. Note that for each $measurement \in E$, $(measurement.q_i == SWAP.q_1) + (measurement.q_i == SWAP.q_2)$ will never be 2 since q_1 and q_2 are two different qubits, and q_i can't be the same qubit with q_1 and q_2 at the same time.

The full heuristic cost function is shown below.

$$H = \max\big(decay(SWAP.q_1), decay(SWAP.q_2)\big) * \left\{\frac{1}{|F|}\sum_{gate \in F} D\big[\pi[gate.q_1]\big]\big[\pi[gate.q_2]\big]\right\} + W * \frac{1}{|E|}\sum_{gate \in E} D\big[\pi[gate.q_1]\big]\big[\pi[gate.q_2]\big] + c * \sum_{measurement \in E}\big((measurement.q_i == SWAP.q_1) + (measurement.q_i == SWAP.q_2)\big) * M[\pi(measurement.q_i)]$$

3.1.4 Reducing Measurement Error by Using SWAPs After the Execution of the Program

For quantum programs, most measurements occur after all operations are done. This is because measurements will destroy the states of the qubits that are used to encode information. Hence, it's very likely that a logical qubit is mapped to a physical qubit with high measurement error at the end of execution due to too many measurements having to occur at this time, and it might be impossible to map all logical qubits waiting for measurements to physical qubits with low measurement errors at the same time. However, unlike quantum gates, where order matters, we can perform measurements in any order. So, we can measure logical qubits that are already being mapped to physical qubits with low measurement error and then use SWAP to move the rest of the logical qubits from the physical qubits with high measurement error to qubits with low measurement error before performing measurements. The following algorithm can be used to address this problem.

```
Algorithm 2: SWAP-based Measurement Error Minimization Algorithm
Input: current mapping $\pi$ $and$ $\pi^{-1}$, measurement error of physical qubits $M$[], Distance matrix D[][], shortest path matrix P[][] (obtained from Dijkstra's algorithm), Quantum circuit QC
set L = []
set N = length of $M$
for each qubit $q_i$ in QC:
    if $q_i$ needs to be measured:
        add $q_i$ to L
sort L ascending order using $M[\pi^{-1}(q_i)]$ for all $q_i$ in L
for $q_i$ in L:
    set min_value = INFINITY
    set target = NONE
    for $0 \le j \le N-1$:
        temp = $1-(1-M[j])(1-D[\pi^{-1}(q_i)][j])$
        if min_value > temp:
            min_value = temp
            Target = $Q_j$ //a physical qubit
    if min_value ≥ $M[\pi^{-1}(q_i)]$:
        measure $\pi^{-1}(q_i)$
    else:
        Using SWAPs move $q_i$ to $Q_j$ following the path
                    $P[\pi^{-1}(q_i)][j]$
        measure $Q_j$
        Update $\pi, \pi^{-1}$
```

The algorithm above will check for all possible options for qubits that need to be measured. There are only two options for measuring a qubit: immediate measurement or moving the logical qubit to another physical qubit and then performing measurements. So, we will first search for a path that has the least error rate overall. Then, we compare this option with immediate measurement and choose the one with the least error rate. The complexity of this algorithm is $O(n^2)$ where n is the number of physical qubits in the quantum device. In the worst-case scenario, all qubits need to be measured. Then, for each qubit, we need to check for all other qubits to find the best swap option, which takes $O(n)$ time. This procedure will repeat n times so the overall complexity is $O(n^2)$.

3.2 Evaluation

3.2.1 Fidelity

Due to time constrain, this algorithm is not implemented. But we can still analyze the algorithm using a theoretical example.

Consider the following quantum circuit and chip coupling graph:

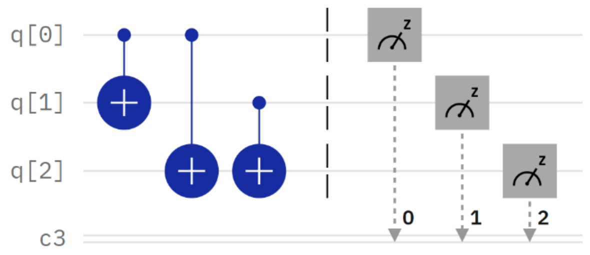


Fig. 11 Example Quantum Circuit

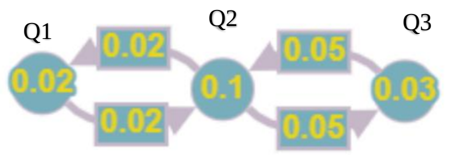


Fig. 12 Example Chip Coupling Graph

In the above chip coupling graph, $Q1, Q2, Q3$ are three physical qubits. The weight on each edge represents the error rate of performing a two-qubit operation between the two qubits connected by the edge. Numbers on each qubit represent the measurement error rate of that qubit.

If we use an algorithm that is not variability-aware, like the original SABRE, then a possible initial mapping is $q_0 \to Q2, q_1 \to Q1, q_2 \to Q3$ (or $q_0 \to Q2, q_1 \to Q3, q_2 \to Q2$). Then, we need to apply CNOTs on Q2 and Q1 first, and then, Q2 and Q3. After that, a SWAP has to be applied to either swap the pair Q1 and Q2 or Q3 and Q2. Finally, three measurements will perform immediately. Based on this, we can calculate the error rate for these operations. In the first step, we apply a CNOT on Q2 and Q1 and another one on Q2 and Q3. Then, the error rate at this time is $1 - (1 - 0.02)(1 - 0.05) = 0.069$. In the second step, different SWAP will have different error rates. If we apply SWAP on Q1 and Q2, the error rate is $1 - (1 - 0.069)(1 - 0.02)^3 = 0.123750248$. If we apply SWAP on Q2 and Q3, the error rate is $1 - (1 - 0.069)(1 - 0.05)^3 = 0.201783875$. Lastly, after three measurements, we will have an error rate of $1 - (1 - 0.123750248)(1 - 0.02)(1 - 0.03)(1 - 0.1) \approx 0.25$ in the best-case scenario, and $1 - (1 - 0.201783875)(1 - 0.02)(1 - 0.03)(1 - 0.1) = 0.32$ in the worst-case scenario.

For my variation-aware SABRE algorithm, the initial mapping will be the same. The difference occurs when we need to swap one of the three qubits. Since the connection between Q1 and Q2 is stronger, we are guaranteed that algorithm will apply SWAP on Q1 and Q2 instead of Q3 and Q2. For the three measurements, the variation-aware SABRE will first measure Q1 and Q3 due to low measurement errors on these two physical qubits. For Q2, the algorithm will choose to apply a SWAP between Q1 and Q2 and measure Q1 instead. Hence, we can calculate the error rate as follows: before we perform the measurements, the error rate is 0.123750248 as calculated above. For the first two measurements, the error rate is $1 - (1 - 0.123750248)(1 - 0.02)(1 - 0.03) = 0.16703698574$. For the last measurement, we will apply a SWAP on Q1 and Q2 and a measurement on Q1, so the final error rate is $1 - (1 - 0.16703698574)(1 - 0.02)^3(1 - 0.02) \approx 0.23$. This is better than the original SABRE algorithm, even in the best-case scenario. In the worst-case scenario of the original SABRE, the error rate is reduced by nearly one-third. This is a significant improvement in terms of fidelity.

3.2.2 Runtime

Now, we will consider the runtime of this algorithm. The original SABRE algorithm has a runtime at most $O(n^{2.5})$ where n is the number of physical qubits. The variability-aware heuristic search requires a few additional steps. Firstly, it needs to compute the SWAP distance, which requires $O(n)$ time because physical qubits can only connect with their neighbors, so the number of edges in the chip coupling graph is $O(n)$. Then, we will use Dijkstra's algorithm to compute the distance matrix with a time complexity $O(n^2)$ in total. Note that the complexity of Dijkstra's algorithm is $O(n^2)$ for a single search in the general case, but we know that the chip coupling graph only has $O(n)$ edges, and Dijkstra only visit each edge one time, so we are guaranteed to have $O(n)$ for each search in our case. We need n searches to find all shortest paths, so the overall complexity is $O(n^2)$. The complexity of VQA is $O(n)$ for the same reason. The complexity of computing the modified heuristic search function will be the same as the original cost function with time complexity $O(n)$ when all qubits appear in F. Lastly, we add a SWAP-based Measurement Error Minimization Algorithm to SABRE. The time complexity of this algorithm is $O(n^2)$ when all qubits need to be measured. So, the time complexity of the variability-aware SABRE will also have a time complexity at most $O(n^{2.5})$.

4. CONCLUSION

In the previous section, we saw that variability-aware policies had improved the fidelity of the quantum program without a significant influence on running time. As long as the connections between physical qubits are not the same and physical qubits have different measurement errors, it's always worth modifying the non-variability-aware qubit mapping algorithm to make them variability-aware.

Moreover, under certain conditions, we can simplify the function that is used to calculate the error rate of SWAPs. Currently, the error rate of CNOT is around 1%, and this implies the error rate of SWAP is around 3%, which is roughly 3 times of error rate of the CNOT gate. When the error rate of the CNOT gate is low, we can obtain the error rate of SWAP by simply multiplying the error rate of the CNOT gate by three. This is because $1 - \left(1 - error(CNOT)\right)^3 = 3 * error(CNOT) - 3\left(error(CNOT)\right)^2 + \left(error(CNOT)\right)^3 \approx 3 * error(CNOT)$ when $error(CNOT)$ is small enough so that we can safely ignore the square and cube terms. Furthermore, if the chip coupling graph of a quantum device has a small diameter (the maximum length of shortest paths for all pairs of nodes), we can use ordinary Dijkstra's algorithm to approximate the error rate of applying a sequence of SWAPs between qubits. This simplification works because the error rate of a sequence of SWAP is $1 - \prod_{k=1}^{N}(1 - error_k(SWAP))$ where N is the number SWAPs, and $error_k(SWAP)$ is the error rate of kth SWAP in the sequence. Then, $1 - \prod_{k=1}^{N}(1 - error_k(SWAP)) = \sum_{k=1}^{N} error_k(SWAP) + \sum_{j=1}^{N} \sum_{i<j} error_i(SWAP) error_j(SWAP) + \sum_{k=1}^{N} \sum_{j<k} \sum_{i<j} error_i(SWAP) error_j(SWAP) error_k(SWAP) + \cdots$. Since N and $error_k(SWAP) \approx 3 * error_k(CNOT)$ are small, we have $1 - \prod_{k=1}^{N}(1 - error_k(SWAP)) \approx \sum_{k=1}^{N} error_k(SWAP) = 3 \sum_{k=1}^{N} error_k(CNOT)$, which is 3 times of the sum of weights in the chip coupling graph.

5. FURTURE WORK

This work is not implemented, so it's not tested on a real quantum device. So, one of the future directions is to implement this algorithm and compare the result with the original SABRE using some benchmark.

Also, this paper only considers gate error and measurement error, while in reality, there are other types of errors in quantum computers. We also assume that errors happen independently when calculating the error rate of applying multiple SWAP or CNOT gates. This simplification might make the distance matrix can't reflect the real strength of connections between physical qubits, which can hinder the performance of the algorithm.

6. ACKNOWLEDGMENTS

This paper is Shuohao Ping's project 2 for CS 515 (16:198:515) Programming Languages And Compilers I at Rutgers University. This project is supervised by Professor Zheng (Eddy) Zhang.

7. REFERENCES

[1] Li, G., Ding, Y., &amp; Xie, Y. (2019, May 7). Tackling the qubit mapping problem for NISQ-era quantum devices. arXiv.org. https://arxiv.org/abs/1809.02573

[2] Chi Zhang, Ari B. Hayes, Longfei Qiu, Yuwei Jin, Yanhao Chen, and Eddy Z. Zhang. 2021. Time-optimal Qubit mapping. In Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '21). Association for Computing Machinery, New York, NY, USA, 360–374. https://doi.org/10.1145/3445814.3446706

[3] Swamit S. Tannu and Moinuddin K. Qureshi. 2019. 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). Association for Computing Machinery, New York, NY, USA, 987–999. https://doi.org/10.1145/3297858.3304007

[4] Das, P., Tannu, S., &amp; Qureshi, M. (2021, September 11). Jigsaw: Boosting Fidelity of NISQ programs via measurement subsetting. arXiv.org. https://arxiv.org/abs/2109.05314

[5] IBM unveils 400 qubit-plus quantum processor and next-generation IBM Quantum System Two. IBM Newsroom. (n.d.). https://newsroom.ibm.com/2022-11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two

[6] Marcos Yukio Siraichi, Vinícius Fernandes dos Santos, Sylvain Collange, and Fernando Magno Quintao Pereira. Qubit allocation. In CGO 2018, pages 113–125, New York, NY, USA, 2018. ACM.

[7] IBM. IBM Q Experience Device. https://quantumexperience.ng.bluemix.net/qx/devices, 2018.

[8] Nation, P., &amp; Johnson, B. (2021, February 22). How to measure and reset a qubit in the middle of a circuit execution. IBM Research Blog. https://www.ibm.com/blogs/research/2021/02/quantum-mid-circuit-measurement/