Enabling multi-programming mechanism for quantum computing in the NISQ era

Siyuan Niu, Aida Todri-Sanial

Enabling multi-programming mechanism for quantum computing in the NISQ era

Siyuan Niu
LIRMM, University of Montpellier
34095 Montpellier, France
siyuan.niu@lirmm.fr

Aida Todri-Sanial
LIRMM, University of Montpellier, CNRS
34095 Montpellier, France
aida.todri@lirmm.fr

ABSTRACT

As NISQ devices have several physical limitations and unavoidable noisy quantum operations, only small circuits can be executed on a quantum machine to get reliable results. This leads to the quantum hardware under-utilization issue. Here, we address this problem and improve the quantum hardware throughput by proposing a multi-programming approach to execute multiple quantum circuits on quantum hardware simultaneously. We first introduce a parallelism manager to select an appropriate number of circuits to be executed at the same time. Second, we present two different qubit partitioning algorithms to allocate reliable partitions to multiple circuits – a greedy and a heuristic. Third, we use the Simultaneous Randomized Benchmarking protocol to characterize the crosstalk properties and consider them in the qubit partition process to avoid crosstalk effect during simultaneous executions. Finally, we enhance the mapping transition algorithm to make circuits executable on hardware using a decreased number of inserted gates. We demonstrate the performance of our multi-programming approach by executing circuits of different sizes on IBM quantum hardware simultaneously. We also investigate this method on VQE algorithm to reduce its overhead.

1 INTRODUCTION

Quantum computing promises to achieve an exponential speedup to tackle certain computational tasks compared with the classical computers [11, 18, 21, 32, 33, 46, 47]. Although quantum technologies are continuously improving, current quantum devices are still qualified as Noisy Intermediate-Scale Quantum (NISQ) hardware [41], with several physical constraints. For example, for superconducting devices which we target in this paper, connections are only allowed between two neighbouring qubits. Besides, the gate operations of NISQ devices are noisy and have unavoidable error rates. As we do not have enough number of qubits to realize Quantum Error Correction [9, 10, 22], only small circuits with limited depth can obtain reliable results when executed on quantum hardware, which leads to the waste of hardware resource. Moreover, with the growing demand to access to quantum hardware, its under-utilization issue increases the waiting time for users, which indicates the need to improve the hardware throughput.

As the qubit number of the hardware increases and the error rates improve, it becomes possible to execute multiple circuits on a quantum chip simultaneously. The multi-programming mapping problem was firstly introduced by [15], which demonstrated that the throughput and utilization of NISQ hardware can be enhanced by executing several circuits at the same time. Ref [16] further improved it in terms of fidelity and gate number by proposing a Community Detection Assisted Partition algorithm along with the X-SWAP scheme (we refer to this algorithm as CDAP for brevity). However, their results showed that when executing multiple quantum circuits simultaneously, the activity of one circuit can negatively impact the fidelity of others, due to the difficulty of allocating reliable regions to each circuit, higher chance of crosstalk error [45], and the qubit movement limitation (only inside of the partition). Previous works [15, 16] have left these issues largely unexplored and have not addressed the problem holistically: (1) Hardware topology and calibration data are not fully analyzed where allocation is done on unreliable or sparse-connected partitions to circuits ignoring the robust qubits and links. (2) These works use only `SWAP` gate for mapping transition process and the modified circuits always have a large number of additional gates. (3) Crosstalk error is not considered when allocating partitions for circuits. For example, the X-SWAP scheme [16] for reducing the inserted `SWAP` number can only be performed when the two circuits are allocated to neighbouring partitions, which can introduce crosstalk effect and decrease the circuit output fidelity. Detrimental crosstalk impact when executing multiple parallel instructions has been reported in [5, 6, 37] by using Simultaneous Randomized Benchmarking (SRB) [23]. In presence of crosstalk, gate error can be increased by an order of magnitude. Ref [5] even proposed a fault-attack model using crosstalk in a multi-programming environment.

It is important to investigate the multi-programming approach in the NISQ era especially for Variational Quantum Algorithms (VQAs) [12]. For example, the multi-programming mechanism can enable to execute several ansatz states in parallel in one quantum processor, such as in Variational Quantum Eigensolver (VQE) [31, 40], Variational Quantum Linear Solver (VQLS) [8, 29], or Variational Quantum Classifier (VQC) [27, 43] with reliability. It is also general enough to be applied to other quantum circuits regardless of applications or algorithms.

In this work, we address the problem of multi-programming while considering the impact of hardware topology, calibration data, and crosstalk without losing the circuit fidelity. First, we introduce a parallelism manager that can optimally select the number of circuits being executed on the quantum hardware simultaneously. Second, we present two different qubit partition algorithms to allocate reliable partitions to different circuits. One is a greedy partition algorithm which provides optimal choices. The other one is based on a heuristic which can give nearly optimal results and significantly reduce the time complexity. Third, we consider crosstalk error during the partition process to lower the crosstalk effect during simultaneous executions. Then, we improve the mapping transition step of the qubit mapping problem to make quantum circuits executable on quantum hardware with a reduced number of additional gates. Finally, we evaluate our algorithm on real quantum hardware by first executing circuits of different sizes at the

same time and then applying it to VQE algorithm to estimate the ground state energy of deuteron. To the best of our knowledge, this is the first attempt to propose a complete multi-programming process flow for executing an optimal number of workloads in parallel ensuring the output fidelity by analyzing the hardware limitations.

2 RESULTS

2.1 Multi-programming workflow

The multi-programming workflow is schematically shown in Fig. 1, which includes the following steps:

- Input layer. It contains a list of small quantum circuits written in OpenQASM language [14], and the quantum hardware information, including the hardware topology, calibration data, and crosstalk effect.
- Parallelism manager. It can determine whether executing circuits concurrently or separately. If the simultaneous execution is allowed, it can further decide the number of circuits to be executed on the hardware at the same time without losing fidelity based on the fidelity metric included in the hardware-aware multi-programming compiler.
- Hardware-aware multi-programming Compiler. Qubits are partitioned to several reliable regions and are allocated to different quantum circuits using qubit partition algorithms. Then, the partition fidelity is evaluated by the post qubit partition process. We introduce a fidelity metric here which helps to decide whether this number of circuits can be executed simultaneously or the number needs to be reduced based on their properties.
- Scheduler. The mapping transition algorithm is applied and circuits are transpiled to be executable on real quantum hardware.
- Output layer. Output circuits are executed on the quantum hardware simultaneously or independently according to the previous steps and the experimental results are obtained.

2.2 Parallelism manager

In order to determine the optimal number of circuits that can be executed on the hardware in parallel without losing fidelity, here, we introduce the parallelism manager, shown in Fig. 2a.

Suppose we have a list of n circuit workloads with n_i qubits for each of them, that are expected to be executed on N-qubit hardware. Firstly, the circuits are sorted according to their densities. The density of a circuit is defined as the number of CNOTs divided by the qubit number of the circuit, $\#CNOTs/n_i$, [16]. Then, we pick K circuits which is the maximum number of circuits that are able to be executed on the hardware at the same time, $\sum_{n=1}^{K} n_i \leq N$. If K is equal to one, then all the circuits should be executed independently. Otherwise, these circuits are passed to the hardware-aware multi-programming compiler. They work together to decide an optimal number of simultaneous circuits to be executed.

2.3 Hardware-aware multi-programming compiler

2.3.1 *Qubit partition.*

Here, we present the key features of the qubit partition algorithms. A motivational example can be found in Supplementary Note 2.

Crosstalk effect characterization.

Crosstalk is one of the major noise sources in NISQ devices, which can corrupt a quantum state due to quantum operations on other qubits [44]. There are two types of crosstalk. The first one is quantum crosstalk, which is caused by the always-on-ZZ interaction [35, 52]. The second one is classical crosstalk caused by the incorrect control of the qubits. The calibration data provided by IBM do not include the crosstalk error. To consider the crosstalk effect in partition algorithms, we must first characterize it in the hardware. There are several protocols presented in [7, 19, 23, 28, 42] to benchmark the crosstalk effect in quantum devices. In this paper, we choose the mostly used protocol – Simultaneous Randomized Benchmarking (SRB) [23] to detect and quantify the crosstalk between CNOT pairs when executing them in parallel.

We characterize the crosstalk effect followed by the optimization methods presented in [37]. On IBM quantum devices, the crosstalk effect is significant only at one hop distance between CNOT pairs [37], such as $(CX_{0,1}|CX_{2,3})$ shown in Fig. 3a, when the control pulse of one qubit propagates an unwanted drive to the nearby qubits that have similar resonate frequencies. Therefore, we perform SRB only on CNOT pairs that are separated by one-hop distance. For those pairs whose distance is greater than one hop, the crosstalk effects are very weak and we ignore them. It allows us to parallelize SRB experiments of multiple CNOT pairs when they are separated by two or more hops. For example, in IBM Q 27 Toronto (ibmq_toronto) [1], the pairs $(CX_{0,1}|CX_{4,7})$, $(CX_{12,15}|CX_{17,18})$, $(CX_{5,8}|CX_{11,14})$ can be characterized in parallel.

We perform the crosstalk characterization on IBM Q 27 Toronto twice. The results show that, although the absolute gate errors vary every day, the pairs that have strong crosstalk effect remain the same across days. SRB experiment on CNOT pairs $(g_i|g_j)$ gives error rate $E(g_i|g_j)$ and $E(g_j|g_i)$. Here, $E(g_i|g_j)$ represents the CNOT error rate of g_i when g_i and g_j are executed in parallel. If there is a crosstalk effect between the two pairs, it will lead to $E(g_i|g_j) > E(g_i)$ or $E(g_j|g_i) > E(g_j)$. The crosstalk effect characterization is expensive and time costly. Some of the pairs do not have crosstalk effect whereas the CNOT error of the pair affected the most by crosstalk effect is increased by more than five times. Therefore, we extract the pairs with significant crosstalk effect, i.e., $E(g_i|g_j) > 3 \times E(g_i)$ and only characterize these pairs when crosstalk properties are needed. We choose the same factor 3 to quantify the pairs with strong crosstalk error like [37]. The result of crosstalk effect characterization on IBM Q 27 Toronto is shown in Fig. 3b.

Greedy sub-graph partition algorithm.

We develop a Greedy Sub-graph Partition algorithm (GSP) for qubit partition process which is able to provide theoretically the optimal partitions for different quantum circuits (see Supplementary Note 3 for pseudo-code of GSP). The first step of the GSP algorithm is to traverse the overall hardware to find all the possible partitions for

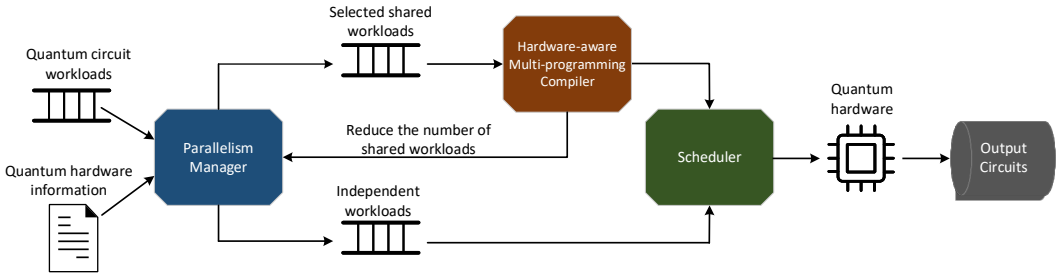


Figure 1: Overview of the proposed multi-programming framework. The input layer includes the quantum hardware information and multiple quantum circuit workloads. The parallelism manager helps to decide whether executing circuits simultaneously or independently. For simultaneous executions, it works with the hardware-aware multi-programming compiler to select an optimal number of shared workloads to be executed at the same time. Then, the scheduler makes all the circuits executable on the quantum hardware and we can obtain the results of the output circuits.

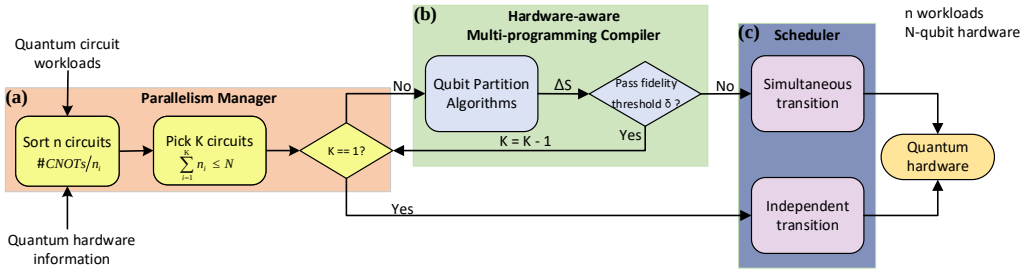


Figure 2: Process flow of each block that constitutes our multi-programming approach. (a) The parallelism manager selects K circuits according to their densities and passes them to the hardware-aware multi-programming compiler. **(b)** The qubit partition algorithms allocate reliable regions to multiple circuits. ΔS is the difference between partition scores when partitioning independently and simultaneously, which is the fidelity metric. δ is the threshold set by the user. The fidelity metric helps to select the optimal number of simultaneous circuits to be executed. **(c)** The scheduler performs mapping transition algorithm and makes quantum circuits executable on real quantum hardware.

a given circuit. For example, suppose we have a five-qubit circuit, we find all the subgraphs of the hardware topology (also called coupling graph) containing five qubits as the partition candidates. Each candidate has a score to represent its fidelity depending on the topology and calibration data. The partition with the best fidelity is selected and all the qubits inside of the partition are marked as used qubits so they cannot be assigned to other circuits. For the next circuit, a subgraph with the required number of qubits is assigned and we check if there is an overlap on this partition to partitions of previous circuits. If not, the subgraph is a partition candidate for the given circuit and the same process is applied to each subsequent circuit. To account for crosstalk, we check if any pairs in a subgraph have strong crosstalk effect caused by the allocated partitions of other circuits. If so, the score of the subgraph is adjusted to take crosstalk error into account.

In order to evaluate the reliability of a partition, there are two factors that need to be considered: partition topology and error rates of two-qubit links and readout error of each qubit. One-qubit gates are ignored for simplicity and because of their relatively low error rates compared to the other quantum operations. If there is a qubit pair in a partition that has strong crosstalk affected by other partitions, then CNOT error of this pair is added to the crosstalk effect. Note that the most recent calibration data should be retrieved through the IBM Quantum Experience before each usage to ensure that the algorithm has access to the most accurate and up-to-date information. To evaluate the partition topology, we determine the longest shortest path (also called graph diameter) of the partition, denoted L. The smaller the longest shortest path is, the better the partition is connected and eventually fewer SWAP gates would be needed to make a connection between two qubits in a well-connected partition.

We devise a fidelity score metric for a partition that is the sum of the graph diameter L, average CNOT error rate of the links times the number of CNOTs of the circuit, and the sum of the readout error rate of each qubit in a partition (Eq. 1). Note that the CNOT error rate includes the crosstalk effect if it exists.

$$Score_g = L + Avg_{CNOT} \times \#CNOTs + \sum_{Q_i \in P} R_{Q_i} \quad (1)$$

The graph diameter L is always prioritized in this equation, since it is more than one order of magnitude larger than the other two factors. The partition with the smallest fidelity score is selected. It is supposed to have the best connectivity and the lowest error rate. Moreover, the partition algorithm prioritizes the quantum circuit with a large density because the input circuits are ordered by their

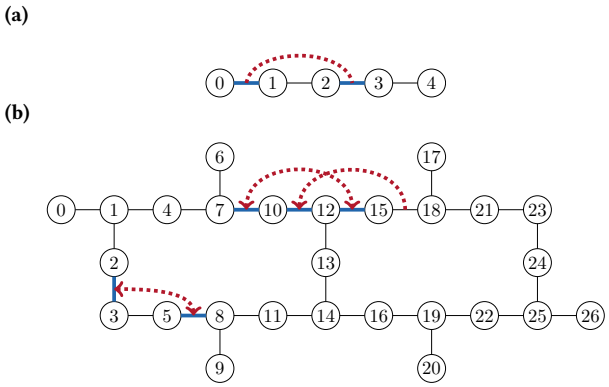


Figure 3: Characterization of crosstalk effect. (a) Crosstalk pairs separated by one-hop distance. The crosstalk pairs should be able to be executed at the same time. Therefore, they cannot share the same qubit. One-hop is the minimum distance between crosstalk pairs. **(b)** Crosstalk effect results of IBM Q 27 Toronto using SRB. The arrow of the red dash line points to the CNOT pair that is affected significantly by crosstalk effect, e.g., $CX_{2,3}$ and $CX_{5,8}$ affect each other when they are executed simultaneously. In our experiments, $E(CX_{10,12}|CX_{15,18}) > 3 \times E(CX_{10,12})$, whereas $E(CX_{15,18}|CX_{10,12}) \approx 2.2 \times E(CX_{15,18})$. As we choose 3 as the factor to pick up pairs with strong crosstalk effect, there is no arrow at pair $CX_{15,18}$.

densities during the parallelism manager process. The partition algorithm is then called for each circuit in order. However, GSP algorithm is expensive and time costly. For small circuits, GSP algorithm gives the best choice of partition. It is also useful to use it as a baseline to compare with other partition algorithms. For beyond NISQ, a better approach should be explored to overcome the complexity overhead.

Qubit fidelity degree-based heuristic sub-graph partition algorithm.

The Qubit fidelity degree-based Heuristic Sub-graph Partition algorithm (QHSP) should perform as well as GSP but without the large runtime overhead.

In QHSP, when allocating partitions, we favor qubits with high fidelity. We define the fidelity degree of qubit based on the CNOT and readout fidelities of this qubit as in Eq. 2.

$$F\_Degree_{Q_i} = \sum_{Q_j \in N(Q_i)} \lambda \times (1 - E[Q_i][Q_j]) + (1 - R_{Q_i}) \quad (2)$$

Q_j are the neighbour qubits connected to Q_i, E is the CNOT error matrix, and R is the readout error rate. λ is a user defined parameter to weight between the CNOT error rate and readout error rate. Such parameter is useful for two reasons: (1) Typically, in a quantum circuit, the number of CNOT operations is different from the number of measurement operations. Hence, the user can decide on λ based on the relative number of operations. (2) For some qubits, the readout error rate is one or more orders of magnitude larger than the CNOT error rate. Thus, it is reasonable to add a weight parameter.

The fidelity degree metric reveals two aspects of the qubit. The first one is the connectivity of the qubit. The more neighbours a qubit has, the larger its fidelity degree is. The second one is the reliability of the qubit accounting CNOT and readout error rates. Thus, the metric allows us to select a reliable qubit with good connectivity. Instead of trying all the possible subgraph combinations (as in GSP algorithm), we propose a QHSP algorithm to build partitions that contain qubits with high fidelity degree while significantly reducing runtime.

To further improve the algorithm, we construct a list of qubits with good connectivity as starting points. We sort all physical qubits (qubits used in hardware) by their physical node degree, which is defined as the number of links in a physical qubit. Note that, the physical node degree is different from the fidelity degree. Similarly, we also obtain the largest logical node degree of the logical qubit (qubits used in the quantum circuit) by checking the number of different qubits that are connected to a qubit through CNOT operations. Next, we compare these two metrics.

If the largest physical node degree is less than the largest logical node degree, it means we cannot find a suitable physical qubit to map the logical qubit with the largest logical node degree that satisfies all the connections. In this case, we only collect the physical qubits with the largest physical node degree. Otherwise, the physical qubits whose physical node degree is greater than or equal to the largest logical node degree are collected as starting points. By limiting the starting points, this heuristic partition algorithm becomes even faster.

For each qubit in the starting points list, it explores its neighbours and finds the neighbour qubit with the highest fidelity degree calculated in Eq. 2, and merges it into the sub-partition. Then, the qubit inside of the sub-partition with the highest fidelity degree explores its neighbour qubits and merges the best one. The process is repeated until the number of qubits inside of the sub-partition is equal to the number of qubits needed. This sub-partition is considered as a subgraph and is added to the partition candidates (see Supplementary Note 3 for pseudo-code of QHSP).

After obtaining all the partition candidates, we compute the fidelity score for each of them. As we start from a qubit with high physical node degree and merge to neighbour qubits with high fidelity degree, the constructed partition is supposed to be well-connected, hence, we do not need to check the connectivity of the partition using the longest shortest path L as in Eq. 1, GSP algorithm. We can only compare the error rates. The fidelity score metric is simplified by only calculating the CNOT and readout error rates as in Eq. 3. It is calculated for each partition candidate and the best one is selected. See supplementary note 3 for an example of explaining QHSP in detail.

$$Score_h = Avg_{CNOT} \times \#CNOTs + \sum_{Q_i \in P} R_{Q_i} \quad (3)$$

Runtime analysis

Let n be the number of hardware qubits, k the number of circuit qubits to be allocated in a partition, g the number of gates that the circuit has.

For GSP algorithm, in most cases, the number of circuit qubits is less than the number of hardware qubits, thus the time cost is $O(k^3 n^k)$. It increases exponentially as the number of circuit

qubits augments. QHSP algorithm starts by collecting a list of m starting points where $m \leq n$. It takes $O(mk^2 + nlog(n) + g)$, which is polynomial. For the detailed explanation of runtime analysis, see Supplementary Note 3.

2.3.2 Post qubit partition.

By default multi-programming mechanism reduces circuit fidelity compared to standalone circuit execution mode. If the fidelity reduction is significant, circuits should be executed independently or the number of simultaneous circuits should be reduced even though the hardware throughput can be decreased as well. Therefore, we consistently check the circuit fidelity difference between independent versus concurrent execution.

We start with qubit partition process for each circuit independently and obtain the fidelity score of the partition. Next, this qubit partition process is applied to these circuits to compute the fidelity score when executing them simultaneously. The difference between the fidelity scores is denoted ΔS, which is the fidelity metric. If ΔS is less than a specific threshold δ, it means simultaneous circuit execution does not detriment significantly the fidelity score, thus circuits can be executed concurrently, otherwise, independently or reduce the number of simultaneous circuits. The fidelity metric along with the parallelism manager help to define the optimal number of simultaneous circuits to be executed.

2.4 Scheduler

2.4.1 Mapping transition algorithm.

The circuits need to be transformed to be executable on real quantum hardware, which includes two steps: initial mapping and mapping transition. The initial mapping of each circuit is created while taking into account swap error rate and swap distance to perform qubit movement operations [39]. The initial mapping of the simultaneous mapping transition process is obtained by merging the initial mapping of each circuit according to its partition. We further improve the mapping transition algorithm [39] by modifying the heuristic cost function to better select the inserted gate. We also introduce the `Bridge` gate to the simultaneous mapping transition process for multi-programming.

First, each quantum circuit is transformed into a more convenient format – Directed Acyclic Graph (DAG) circuit which represents the operation dependencies of the circuit without considering the connectivity constraints. Then, the compiler traverses the DAG circuit and goes through each quantum gate sequentially. The gate that does not depend on other gates (i.e., all the gates before it have been executed) is allocated to the first layer, denoted F. The compiler checks if the gates on the first layer are hardware-compliant. The hardware-compliant gates can be executed on the hardware directly without modification. They are added to the scheduler, removed from the first layer and marked as executed. If the first layer is not empty, which means some gates are non-executable on hardware, a `SWAP` or `Bridge` gate is needed. We collect all the possible `SWAPs` and `Bridges`, and use the cost function H (see Eq. 5) to find the best candidate. The process is repeated until all the gates are marked as executed (see Supplementary Note 4 for pseudo-code of simultaneous mapping transition algorithm).

A `SWAP` gate requires three `CNOTs` and inserting a `SWAP` gate can change the current mapping. A `Bridge` gate requires four `CNOTs` and inserting a `Bridge` gate does not change the current mapping and it can only be used to execute a `CNOT` when the distance between the control qubit and the target qubit is exactly two. Both gates need three supplementary `CNOTs`. The `SWAP` gate is preferred when it has a positive impact on the following gates, allocated in the extended layer E, hence it makes these gates executable or reduces the distance between control and target quits. Otherwise, a `Bridge` gate is preferred.

A cost function H is introduced to evaluate the cost of inserting a `SWAP` or `Bridge`. We use the following distance matrix (Eq. 4) as in [39] to quantify the impact of the `SWAP` or `Bridge` gate,

$$D = \alpha_1 \times S + \alpha_2 \times \mathcal{E} \tag{4}$$

where S is the swap distance matrix and $\mathcal{E}$ is the swap error matrix. We set α_1 and α_2 to 0.5 to equally consider the swap distance and swap error rate. In [39], only the impact of a `SWAP` and `Bridge` on other gates (first and extended layer) was considered without considering their impact on the gate itself. As each of them is composed of either three or four `CNOTs`, their impact cannot be ignored. Hence, in our multi-programming mapping transition algorithm, we take self impact into account and create a list of both `SWAP` and `Bridge` candidates, labeled as "tentative gates" and the heuristic cost function is as:

$$H = \frac{1}{|F + N_{Tent}|}\Big(\sum_{g \in F} D[\pi(g.q_1)][\pi(g.q_2)] + \sum_{g \in Tent} D[\pi(g.q_1)][\pi(g.q_2)]\Big) + W \times \frac{1}{|E|} \sum_{g \in E} D[\pi(g.q_1)][\pi(g.q_2)] \tag{5}$$

where W is the parameter that weights the impact of the extended layer, N_{Tent} is the number of gates of the tentative gate, $Tent$ represents a `SWAP` or `Bridge` gate, and π represents the mapping. `SWAP` gate has three `CNOTs`, thus N_{Tent} is three and we consider the impact of three `CNOTs` on the first layer. The mapping is the new mapping after inserting a `SWAP`. For `Bridge` gate, N_{Tent} is four and we consider four `CNOTs` on the first layer, and the mapping is the current mapping as `Bridge` gate does not change the current mapping. We weight the impact on the extended layer to prioritize the first layer. This cost function can help the compiler select the best gate to insert between a `SWAP` and `Bridge` gate.

2.5 Application: simultaneous executions of multiple circuits of different size

2.5.1 Experimental results.

We first evaluated our multi-programming approach by executing a list of different-size benchmarks at the same time on two quantum devices, IBM Q 27 Toronto and IBM Q 65 Manhattan (ibmq_manhattan) [2] (see Supplementary Note 1 for further information about the selected quantum hardware). All the benchmarks are collected from the previous work [54], including several functions taken from RevLib [49] as well as some quantum algorithms written in Quipper [25] or Scaffold [4]. These benchmarks are widely used in the quantum community and their details are

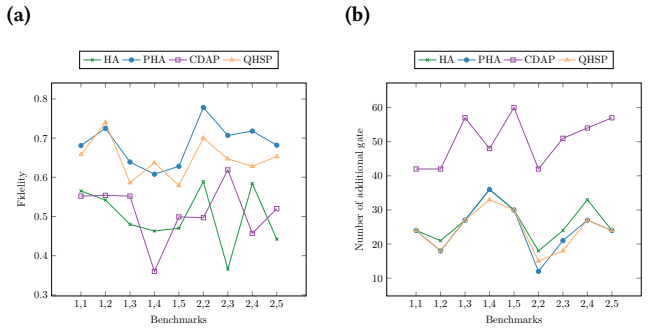


Figure 4: Comparison of fidelity and number of additional gates on IBM Q 27 Toronto when executing two circuits simultaneously. (a) Fidelity**. (b)** Number of additional gates.

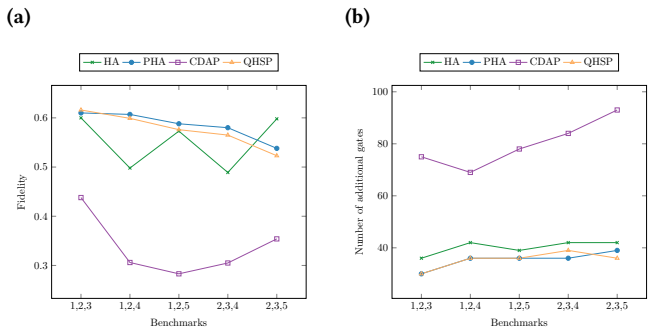


Figure 5: Comparison of fidelity and number of additional gates on IBM Q 65 Manhattan when executing three circuits simultaneously. (a) Fidelity**. (b)** Number of additional gates.

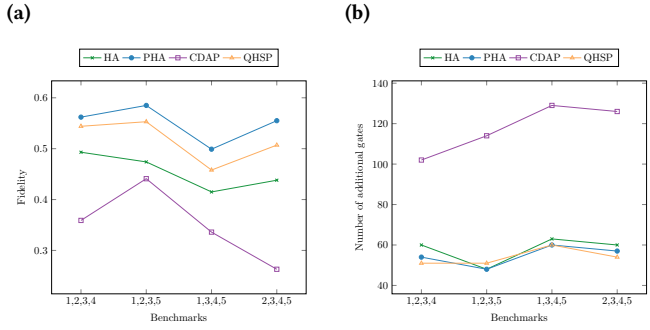


Figure 6: Comparison of fidelity and number of additional gates on IBM Q 65 Manhattan when executing four circuits simultaneously. (a) Fidelity**. (b)** Number of additional gates.

shown in Table 1. We chose small quantum circuits with shallow-depth since only small circuits can obtain reliable results when executed on real quantum hardware. The metrics we used to evaluate our algorithm include Probability of a Successful Trial (PST), number of additional CNOT gates, and Trial Reduction Factor (TRF), see Method for detailed explanation.

Several published qubit mapping algorithms [26, 30, 34, 36, 39, 50, 53] and multi-programming mapping algorithms are available as discussed in section 1. HA [39] seems to be the best qubit mapping algorithm in terms of the number of additional gates and circuit fidelity. We use HA as the baseline for independent executions of multiple circuits. CDAP algorithm proposed in [16] seems to be the best multi-programming mapping algorithm and is considered as the baseline for concurrent executions of multiple circuits.

To summarize, we compare our multi-programming algorithms, 1) GSP + improved mapping transition (labeled as GSP) and 2) QHSP + improved mapping transition (labeled as QHSP), with the baseline CDAP. The loss of fidelity due to simultaneous executions of multiple circuits is reported by comparing concurrent versus independent executions. Moreover, we compare the partition + improved mapping transition algorithm based on HA (labeled as PHA) versus HA on independent executions to show the impact of partition in large quantum hardware for a small circuit. The details of the configuration of algorithms are presented in Methods.

We first ran two quantum circuits on IBM Q 27 Toronto simultaneously. Results on output state fidelity and the number of additional gates are shown in Fig. 4. For independent executions, the fidelity is improved by 46.8% and the number of additional gates is reduced by 8.7% comparing PHA to HA. For simultaneous executions, QHSP and GSP allocate the same partitions except for the first experiment – (ID1, ID1). In this experiment, GSP improves the fidelity by 6% compared to QHSP. Partition results might be different due to the various calibration data and the choice of λ, but the difference of the partition fidelity score between the two algorithms is small. The results show that QHSP is able to allocate nearly optimal partitions while reducing runtime significantly. Therefore, for the rest experiments, we only evaluate QHSP algorithm. QHSP can improve the fidelity by 28.9% and reduce the additional gate number by 52.3% compared to CDAP. Comparing simultaneous (QHSP) versus independent (PHA) executions for two circuits, fidelity decreases by 5.8% and the number of additional gates is almost the same. During the post-partition process, ΔS does not pass the threshold and TRF is two.

Next, we executed on IBM 65 Manhattan three and four simultaneous quantum circuits. Fig. 5 and Fig. 6 show the comparison of fidelity and the number of additional gates. PHA always outperforms HA for independent executions. QSHP significantly outperforms CDAP with the number of simultaneous circuits increasing. The output fidelity is increased by 74.8% and 55.3% on average for the two cases. The reduction of inserted gate number is always more than 50%. The threshold is still not passed and TRF becomes three

ID	Name	Qubits	Num_g	Num_CNOT
1	3_17_13	3	36	17
2	4mod5-v1_22	5	21	11
3	mod5mils_65	5	35	16
4	alu-v0_27	5	36	17
5	decod24-v2_43	4	52	22

Table 1: Information of benchmarks

and four. Moreover, fidelities decrease by 1.5% and 6.7% when comparing simultaneous (QHSP) versus independent (PHA) executions.

Finally, to evaluate the hardware limitations of executing multiple circuits in parallel, we set the threshold δ to 0.2. All the five benchmarks are able to be executed simultaneously on IBM Q 65 Manhattan. Partition fidelity difference is 0.18. Results show that fidelity of simultaneous executions (QHSP) is decreased by 9.5% compared to independent executions (PHA). Both fidelity and additional gate number improvement of QHSP are more than 50% compared to CDAP. The complete experimental results can be found in Supplementary Note 5.

2.5.2 Result analysis.
For independent executions, algorithm PHA is always better than HA due to two reasons: (1) The initial mapping of the two algorithms is based on a random process. During the experiment, we perform the initial mapping generation process ten times and select the best one. However, for PHA, we first limit the random process into a reliable and well-connected small partition space rather than the overall hardware space used by HA. Therefore, with only ten trials, PHA finds a better initial mapping. (2) We improve the mapping transition process of PHA, which can make a better selection between `SWAP` and `Bridge` gate. HA is shown to be sufficient for hardware with a small number of qubits for example a 5-qubit quantum chip. If we want to map a circuit on large hardware, it is better to first limit the search space into a reliable small partition and then find the initial mapping. This qubit partition approach can be applied to general qubit mapping problem for search space limitation when large hardware is selected to map.

For simultaneous executions, QHSP performs better than CDAP because of the following reasons: (1) CDAP constructs a hierarchy tree according to the modularity-based FN community detection algorithm [38]. The tree is constructed by calculating the modularity of the overall hardware coupling graph. However, when allocating a partition to a circuit, we focus on the topology and calibration data inside of the partition, rather than the whole hardware. As the number of partitions to allocate increases, the performance of CDAP becomes worse. (2) CDAP only considers the `SWAP` gate to realize the connection ignoring the `Bridge` gate, which can significantly reduce the number of additional gates. (3) CDAP does not consider the crosstalk effect. Although the X-SWAP scheme used in CDAP can slightly reduce the number of additional gates, it only works when the allocated partitions are close to each other, which will increase the crosstalk effect. However, QHSP takes the partition topology, error rate, and crosstalk effect into consideration and can provide better partitions. QHSP uses almost the same number of additional gates whereas fidelity is decreased less than 10% compared to PHA if the threshold is set to 0.1.

2.6 Application: Estimate the ground state energy of deuteron

In order to demonstrate the potential interest to apply the multi-programming mechanism to existing quantum algorithms, we investigated it on VQE algorithm. To do this, we performed the same experiment as [17, 24] on IBM Q 65 Manhattan, estimating the ground state energy of deuteron, which is the nucleus of a deuterium atom, an isotope of hydrogen.

Deuteron can be modeled using a 2-qubit Hamiltonian spanning four Pauli strings: ZI, IZ, XX, and YY, [17, 24]. If we use the naive measurement to calculate the state energy, one ansatz corresponds to four different measurements. Pauli operator grouping has been proposed to reduce this overhead by utilizing simultaneous measurement [13, 24, 31]. For example, the Pauli strings can be partitioned into two commuting families: $\{ZI, IZ\}$ and $\{XX, YY\}$ using the approach proposed in [24]. It allows one parameterized ansatz to be measured twice instead of four measurements in naive method.

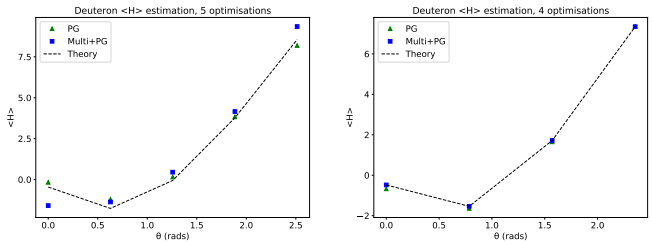
We used a simplified Unitary Coupled Cluster ansatz with a single parameter and three gates, as described in [17, 24]. The algorithm configuration of this experiment is explained in Methods. We applied our multi-programming method on the top of the Pauli operator grouping approach (labeled as PG) [24]. We performed this experiment twice across different days. For the first experiment, the parallelism manager worked with the hardware-aware multi-programming compiler to finally select ten circuits for simultaneous execution without passing the fidelity threshold. It corresponds to perform five optimisations (five different parameterized circuits) at the same time (one parameterized circuit needs two measurements). The selected ten circuits were passed to the scheduler to be executed in parallel. The required circuit number is reduced by ten times compared to PG. Note that, if we use the naive measurement, the number of circuits needed will be reduced by a factor of 20. The result is shown in Fig. 7a. The error rate is quite high for the two executions, 29.7% for PG and 64.4% for multi-programming + PG. The result of the second experiment is shown in Fig. 7b. In this case, four optimisations (eight circuits) were selected to be executed at the same time with respect to the fidelity threshold. The error rate is 9.3% and 7% for the two methods. Applying multi-programming can even improve the output fidelity. The huge fidelity difference is due to the different calibration data of the device which are the input of our multi-programming approach. The complete result of the two experiments including hardware throughput is shown in Fig. 7c.

3 DISCUSSION

In this article, we presented a multi-programming approach that allows to execute multiple circuits on a quantum chip simultaneously without losing fidelity. We introduced the parallelism manager and fidelity metric to select optimally the number of circuits to be executed at the same time. Moreover, we proposed a hardware-aware multi-programming compiler which contains two qubit partition algorithms taking hardware topology, calibration data, and crosstalk effect into account to allocate reliable partitions to different quantum circuits. We also demonstrated an improved simultaneous mapping transition algorithm which helps to transpile the circuits on quantum hardware with a reduced number of inserted gates.

We first executed a list of circuits of different sizes simultaneously and compared our algorithm with the state-of-the-art multi-programming approach. Experimental results showed that our approach can outperform the state of the art in terms of both output fidelity and the number of additional gates. Then, we investigated our multi-programming approach on VQE algorithm to estimate the ground state energy of deuteron, showing the added value of

(a) **(b)**



(c)

Experiments		n_c	Error rate (%)	Hardware throughput
ID1	PG	1	29.7	0.03
	Multi+PG	10	64.4	0.3
ID2	PG	1	9.3	0.03
	Multi+PG	8	7	0.25

Figure 7: The estimation of the ground state energy of deuteron under PG and muti-programming + PG. (a) Five optimisations with ten measurements. **(b)** Four optimisations with eight measurements. **(c)** The complete result of the two experiments. n_c is the number of simultaneous circuit number.

applying our approach to existing quantum algorithms. The multi-programming approach is evaluated on IBM hardware, but it is general enough to be adapted to other quantum hardware.

Based on the experimental result, we found that the main concern with multi-programming mechanism is a trade-off between output fidelity and the hardware throughput. For example, how one can decide which programs to execute simultaneously and how many of them to execute without losing fidelity. Here, we list several guidelines to help the user to utilize our multi-programming approach.

- Check the target hardware topology and calibration data. The multi-programming mechanism is more suitable for a relatively large quantum chip compared to the quantum circuit and with low error rate.
- Choose appropriate fidelity threshold for post qubit partition process. A high threshold can improve the hardware throughput but lead to the reduction of output fidelity. It should be set carefully depending on the size of the benchmark. For benchmarks of small size that we used in experiments, it is reasonable to set the threshold to 0.1.
- The number of circuits that can be executed simultaneously will mainly depend on the fidelity threshold and the calibration data of the hardware.
- QHSP algorithm is suggested for the partition process due to efficiency and GSP is recommended to evaluate the quality of the partition algorithm. Using both algorithms, one can explore which circuits can be executed simultaneously and how many of them within the given fidelity threshold.

Quantum hardware development with more and more qubits will enable execution of multiple quantum programs simultaneously and possibly a linchpin for quantum algorithms requiring parallel sub-problem executions. Variational Quantum Algorithm is becoming a leading strategy to demonstrate quantum advantages for practical applications. In such algorithms, the preparation of parameterized quantum state and the measurement of expectation value are realized on shallow circuits [51]. Taking VQE as an example, the Hamiltonian can be decomposed into several Pauli operators and simultaneous measurement by grouping Pauli operators have been proposed in [13, 24, 31] to reduce the overhead of the algorithm. Based on our experiment, we have shown that the overhead of VQE can be further improved by executing several sets of Pauli operators at the same time using multi-programming mechanism.

For future work, we would like to apply our multi-programming algorithm to other variational quantum algorithms such as VQLS or VQC to enable the preparation of states in parallel and to reduce the overhead of these algorithms. Moreover, in our qubit partition algorithms, we take the crosstalk effects into consideration by characterizing them and adding them to the fidelity score of the partition, which is able to avoid the crosstalk error in a high level. There are some other approaches of eliminating the crosstalk error in a cheaper way instead of performing SRB protocol, for example using commutativity rules to reorder the simultaneous gate operations [30, 37]. However, these methods have some challenges such as trading off between crosstalk and decoherence. More interesting tricks for crosstalk mitigation need to be targeted for simultaneous executions. In addition, not all the benchmarks have the same circuit depth. Taking the time-dependency into consideration, choosing the optimal combination of circuits of different depth to run simultaneously can also be the focus of future work.

4 METHODS

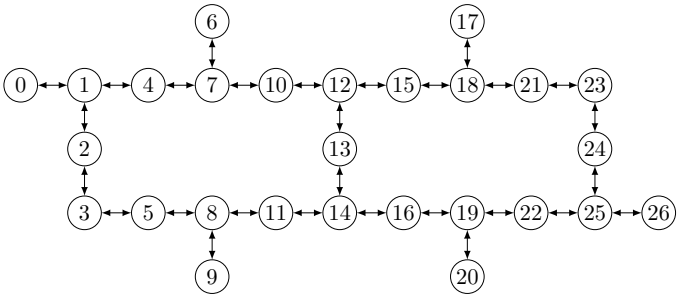
4.1 Metrics

Here are the detailed explanations of the metrics that we use to evaluate our algorithm.

(1) Probability of a Successful Trial (PST) [48]. This metric is defined by the number of trials that give the expected result divided by the total number of trials. The expected result is obtained by executing the quantum circuit on the simulator. To have a precise estimation of the PST, we execute each quantum circuit on the quantum hardware for a large number of trials (8192).
(2) Number of additional `CNOT` gates. This metric is related to the number of `SWAP` or `Bridge` gates inserted. This metric can show the ability of the algorithm to reduce the number of additional gates.
(3) Trial Reduction Factor (TRF). This metric is introduced in [15] to evaluate the improvement of the throughput thanks to the multi-programming mechanism. It is defined as the ratio of trials needed when quantum circuits are executed independently to the trials that when they are executed simultaneously.

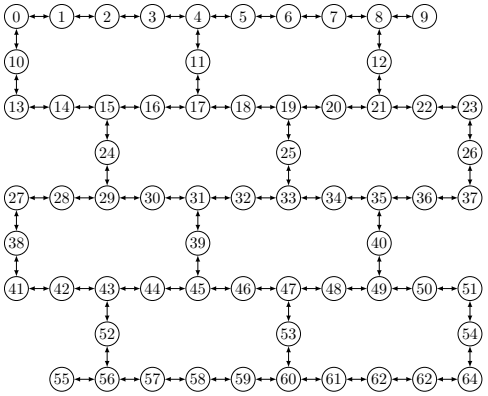
4.2 Algorithm configurations

Here, we consider the algorithm configurations of different multi-programming and standalone mapping approaches. We select the best initial mapping out of ten attempts for HA, PHA, GSP, and QHSP. Weight parameter W in the cost function (Eq. 5) is set to 0.5



Single qubit error rate: 2.168e−4 to 1.455e−3
CNOT error rate: 5.697e−3 to 2.735e−2
Readout error rate: 7.3e−3 to 2.317e−1

Figure 8: IBM Q 27 Toronto topology and error rates.



Single qubit error rate: 2.561e−4 to 5.407e−3
CNOT error rate: 7.619e−3 to 4.638e−2
Readout error rate: 9.2e−3 to 1.118e−1

Figure 9: IBM Q 65 Manhattan topology and calibration data.

and the size of the extended layer is set to 20. Parameters α_1 and α_2 are set to 0.5 respectively to consider equally the swap distance and swap error rate. For the experiments of multiple different-size circuits, the weight parameter λ of QHSP (Eq. 2) is set to 2 because of the relatively large number of CNOT gates in benchmarks, whereas for deuteron experiment, λ is set to 1 because of the small number of CNOTs of the parameterized circuit. The threshold δ for post qubit partition is set to 0.1 to ensure the multi-programming fidelity. Due to the expensive cost of SRB, we perform SRB only on IBM Q 27 Toronto and collect the pairs with significant crosstalk effect. Only the collected pairs are characterized and their crosstalk properties are provided to the partition process. The experimental results on IBM Q 65 Manhattan do not consider the crosstalk effect. For each algorithm, we only evaluate the mapping transition process, which means no optimisation methods like gate commutation or cancellation are applied.

The algorithm is implemented in Python and evaluated on a PC with 1 Intel i5-5300U CPU and 8 GB memory. Operating System is Ubuntu 18.04. All the experiments were performed on the IBM quantum information science kit (qiskit) [20] and the version used is 0.21.0.

5 DATA AVAILABILITY

The source code of the algorithms used in this paper is available on the Github repository [3].

6 SUPPLEMENTARY INFORMATION

6.1 Supplementary note - Hardware information

Noise can cause several errors during the execution process such as (1) coherence errors due to the fragile nature of qubits. The qubit can only maintain information for a limited amount of time. (2) Operational errors including gate errors and measurement errors (readout errors). (3) Crosstalk errors that violate the isolated qubit state due to operations on other qubits.

Supplementary Fig. 8 shows the hardware topology and the calibration data of IBM Q 27 Toronto. We list the calibration data of single-qubit error rate, CNOT error rate, and readout error rate. Note that these errors are not constant and change at each re-calibration of the chip, and IBM does not provide the statistics of crosstalk error. The other device that we choose to evaluate our algorithm is IBM Q 65 Manhattan. Its topology and calibration data are shown in Supplementary Fig. 9. CNOT error rate is one order of magnitude higher than their one-qubit counterparts. Moreover, the readout error rate is of the same order of magnitude or higher than CNOT error rate. In this paper, we only focus on CNOT error rate and readout error rate because of the relatively low error rates of one-qubit gates.

It is important to note that all the interconnects between qubits as well as the reliability of qubit are not equal with respect to CNOT error rate and readout error rate. Taking IBM Q 27 Toronto as an example, the best CNOT gate has an error rate of 4.8 times lower than the worst CNOT, and the most reliable qubit has a readout error rate of 31.7 times lower than the worst qubit. Therefore, each qubit cannot be treated equally, and we need to consider the error difference between the links and qubits.

In this article, we mainly focus on IBM architectures. But the proposed methods are general enough to be applied to any other quantum chips that use the quantum-gate model of computation, such as Google's Sycamore [56] or Rigetti's Aspen-8.

6.2 Supplementary note - Motivational example

To motivate the qubit partition problem, we execute two small circuits QC1 and QC2 simultaneously on IBM Q 27 Toronto with different partitions (Supplementary Fig. 10). CNOT error rate of each link is shown in the figure and the unreliable links and qubits with high readout error rates are highlighted in red. Both circuits have five qubits with a different number of gates as listed in Supplementary Fig. 11.

There are two constraints to be considered when executing multiple circuits concurrently. First, each circuit should be allocated to a partition containing reliable physical qubits. Allocated physical qubits can not be shared among quantum circuits. Second, qubits

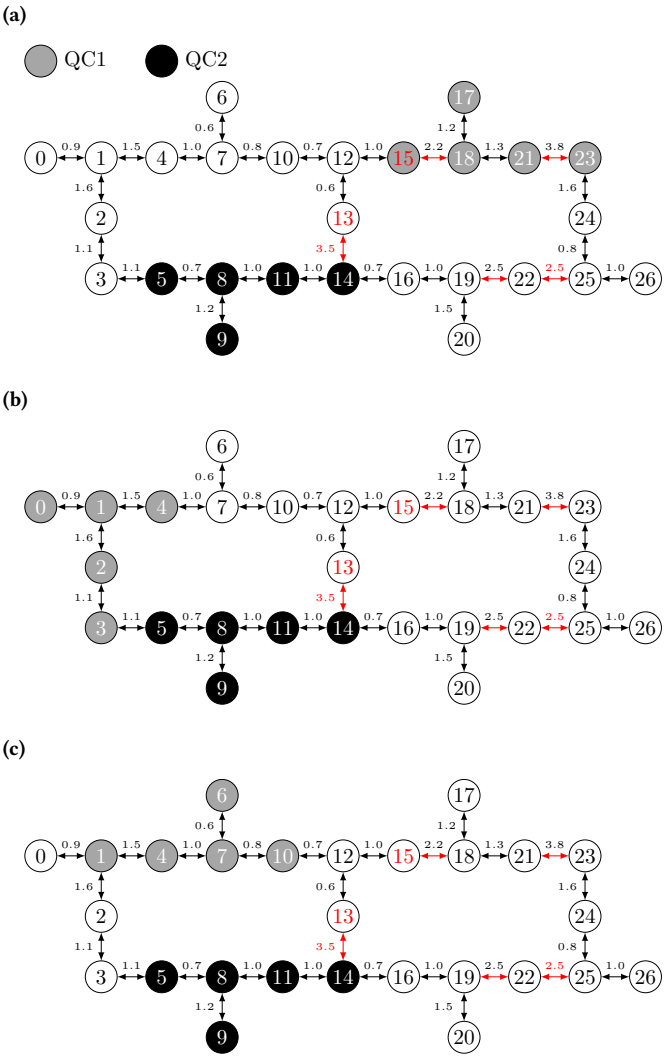


Figure 10: A motivational example of qubit partition problem (error rate in %). (a) Partition without considering operational error. **(b)** Partition considering operational error without considering crosstalk effect. **(c)** Partition considering both operational error and crosstalk effect.

can be moved only inside of their circuit partition, in other words, qubits can be swapped within the same partition only. Thus, finding reliable partitions for multiple circuits is an important step in the multi-programming problem.

We compare three partitions with the same topology to show the impact of different error sources on the output fidelity: (1) Partition P1 without considering the operational error (Supplementary Fig. 10a). (2) Partition P2 only considering operational error without the crosstalk effect (Supplementary Fig. 10b). (3) Partition P3 considering both operational error and crosstalk effect (Supplementary Fig. 10c). Note that the operational error includes CNOT error and readout error. For illustration, we fix the partition of QC2 to

(a)

Benchmarks	name	n	g	P1	P2	P3
QC1	alu-v0_27	5	36	0.256	0.289	0.467
QC2	4mod5-v1_22	5	21	0.792	0.759	0.783

(b)

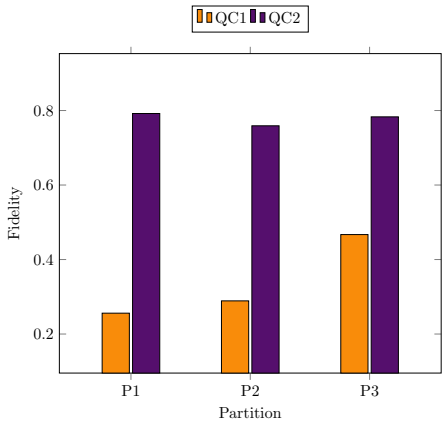


Figure 11: Results of the motivational example. (a) Circuit information and output fidelity results of different partitions. n: qubit number. g: gate number of the circuit. **(b)** Output fidelity results of different partitions.

{5, 8, 9, 11, 14} and only change the partition of QC1. It is important to note that if we have different topologies, the fidelity of the circuit will be different as well because the number of additional gates is strongly related to the hardware topology.

Results in Supplementary Fig. 11b show that Partition P1 has the lowest fidelity. Partition P2 considers operational error and selects {0, 1, 2, 3, 4} with reliable qubits and links. However, it does not consider the crosstalk effect. Since Q_3 is the neighour of Q_5, when $CX_{2,3}$ and $CX_{5,8}$ are executed at the same time, they can affect each other and violate the qubit state. Partition P3 includes {1, 4, 6, 7, 10} and considers both operational error and crosstalk effect. P3 does not have the crosstalk effect and is slightly better than P2 in terms of the operational error, however, the output fidelity of QC1 is increased by 61.6%.

6.3 Supplementary note - Qubit partition

In this note, we first demonstrate the pseudo-code of GSP algorithm. Then, we show an example of QHSP algorithm and its pseudo-code. Finally, we explain the runtime analysis of the two algorithms in detail.

6.3.1 *Greedy sub-graph partition algorithm.*

The pseudo-code of GSP is shown in Algorithm 1.

6.3.2 *Qubit fidelity degree-based heuristic sub-graph partition algorithm.*

Supplementary Fig. 12 shows an example of applying QHSP on IBM Q 5 Valencia (5-qubit ibmq_valencia) [55] for a four-qubit circuit. The calibration data of IBM Q 5 Valencia, including readout error rate and CNOT error rate is shown in Supplementary Fig. 12a. The

Algorithm 1: GSP algorithm

input : Quantum circuit QC, Coupling graph G, Calibration data C, Crosstalk properties crosstalk_props, Used_qubits q_{used}
output : A list of candidate partitions sub_graph_list

```
begin
    qubit_num ← QC.qubit_num;
    Set sub_graph_list to empty list;
    for sub_graph ∈ combinations (G, qubit_num) do
        if sub_graph is connected then
            if q_used is empty then
                sub_graph.Set_Partition_Score (G, C, QC);
                sub_graph_list.append (sub_graph);
            end
            if no qubit in sub_graph is in q_used then
                crosstalk_pairs ← Find_Crosstalk_pairs (sub_graph, crosstalk_props, q_used);
                sub_graph.Set_Partition_Score (G, C, QC, crosstalk_pairs);
                sub_graph_list.append (sub_graph);
            end
        end
    end
    return sub_graph_list;
end
```

Algorithm 2: QHSP algorithm

input : Quantum circuit QC, Coupling graph G, Calibration data C, Crosstalk properties crosstalk_props, Used_qubits q_{used}, Starting points starting_points
output : A list of candidate partitions sub_graph_list

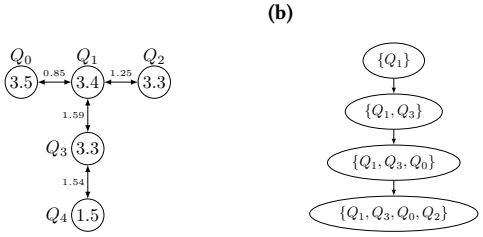
```
begin
    circ_qubit_num ← QC.qubit_num;
    Set sub_graph_list to empty list;
    for i ∈ starting_points do
        Set sub_graph to empty list;
        qubit_num ← 0;
        while qubit_num < circ_qubit_num do
            if sub_graph is empty then
                sub_graph.append (i);
                qubit_num ← qubit_num + 1 ;
                continue;
            end
            best_qubit ← find_best_qubit (sub_graph, G, C);
            if best_qubit ≠ None then
                sub_graph.append (best_qubit);
                qubit_num ← qubit_num + 1 ;
                continue;
            end
        end
        if len (sub_graph) = circ_qubit_num then
            if q_used is empty then
                sub_graph.Set_Partition_Error (G, C, QC,);
                sub_graph_list.append (sub_graph);
            end
            if no qubit in sub_graph is in q_used then
                crosstalk_pairs ← Find_Crosstalk_pairs (sub_graph, crosstalk_props, q_used);
                sub_graph.Set_Partition_Error (G, C, QC, crosstalk_pairs);
                sub_graph_list.append (sub_graph);
            end
        end
    end
    return sub_graph_list;
end
```

fidelity degree of qubit calculated by Eq. 2 is shown in Supplementary Fig. 12c. Here, we consider a circuit of medium size and set λ to two. Suppose the largest logical degree is three. Therefore, Q_1 is selected as the starting point since it is the only physical qubit that has the same physical node degree as the largest logical degree. It has three neighbour qubits: Q_0, Q_2, and Q_3. Q_3 is merged into the sub-partition because it has the highest fidelity degree among neighbour qubits. The sub-partition becomes $\{Q_1, Q_3\}$. As the fidelity degree of Q_1 is larger than Q_3, the algorithm will select again the left neighbour qubit with the largest fidelity degree of Q_1, which is Q_0. The sub-partition becomes $\{Q_1, Q_3, Q_0\}$. Q_1 is still the qubit with the largest fidelity degree in the current sub-partition, its neighbour qubit – Q_2 is merged. The final sub-partition is $\{Q_1, Q_3, Q_0, Q_2\}$ and it can be considered as a partition candidate. The merging process is shown in Supplementary Fig. 12b.

The pseudo-code of QHSP is shown in Algorithm 2.

6.3.3 Runtime analysis.
Let n be the number of hardware qubits and k the number of qubits in the circuit to be allocated in a partition. GSP algorithm selects all the combinations of k subgraphs from n-qubit hardware and takes $O(C(n,k))$ time, which is $O(n\ choose\ k)$. For each subgraph, it computes its fidelity score including calculating the longest shortest path, which scales at $O(k^3)$. It ends up being equivalent to $O(k^3 min(n^k, n^{n-k}))$. In most cases, the number of circuit qubits is less than the number of hardware qubits, thus the time complexity becomes $O(k^3 n^k)$. It increases exponentially as the number of qubits of the circuit augments.

QHSP algorithm starts by collecting a list of m starting points where $m \leq n$. To get the starting points, we sort the n physical qubits by their physical node degree, which takes $O(nlog(n))$. Then, we iterate over all the gates of the circuit (e.g. circuit has g gates) and sort the k logical qubits according to the logical node degree, which



(c)

Qubit	Q_0	Q_1	Q_2	Q_3	Q_4
Fidelity degree	1.96	3.93	1.95	2.94	1.97
Physical node degree	1	3	1	2	1

Figure 12: Example of qubit partition on IBM Q 5 Valencia for a four-qubit circuit using QHSP. Suppose the largest logical degree of the target circuit is three. **(a)** Calibration data of IBM Q 5 Valencia. The value inside of the node represents the readout error rate (in%), and the value above the link represents the CNOT error rate (in%). **(b)** Process of constructing a partition candidate using QHSP. (c) The physical node degree and the fidelity degree of each qubit calculated by Eq. 2.

takes $O(g + klog(k))$. Next, for each starting point, it iteratively merges the best neighbour qubit until each sub-partition contains k qubits. To find the best neighbour qubit, the algorithm finds the best qubit in a sub-partition and traverses all its neighbours to select the one with the highest fidelity degree. Finding the best qubit in the sub-partition is $O(p)$ where p is the number of qubits in a sub-partition. The average number of qubits p is $k/2$, so this process takes $O(k)$ time on average. Finding the best neighbour qubit is $O(1)$ because of the nearest-neighbor connectivity of superconducting devices. Overall, the QHSP takes $O(mk^2 + nlog(n) + g + klog(k))$ time, and it can be truncated to $O(mk^2 + nlog(n) + g)$, which is polynomial.

6.4 Supplementary note - Mapping transition algorithm

In this note, we present the pseudo-code of our simultaneous mapping transition algorithm (see Algorithm 3).

6.5 Supplementary note - Experimental results

In this note, we demonstrate the exact experimental results when executing a different number of circuits on the two devices, IBM Q 27 Toronto and IBM Q 65 Manhattan, at the same time.

Algorithm 3: Simultaneous mapping transition algorithm

input : Circuits *DAGs*, Coupling graph G, Distance matrices Ds, Initial mapping π_i, First layers Fs
output : Final schedule schedule

begin
  $\pi_c \leftarrow \pi_i$;
  while *not all gates are executed* **do**
    Set swap_bridge_lists to empty list;
    for F_i *in* Fs **do**
      for gate *in* F_i **do**
        if gate *is hardware-compliant* **then**
          schedule.append (gate);
          Remove gate from F_i;
        end
      end
      if F_i *is not empty* **then**
        swap_bridge_candidate_list ← FindSwapBridgePairs (F_i, G);
        swap_bridge_lists.append (swap_bridge_candidate_list);
      end
    end
    for swap_bridge_candidate_list ∈ swap_bridge_lists **do**
      for g_{tmp} ∈ swap_bridge_candidate_list **do**
        $\pi_{tmp} \leftarrow$ Map_Update (g_{tmp}, π_c);
        $H_{basic} \leftarrow 0$;
        for gate ∈ F_i **do**
          $H_{basic} \leftarrow H_{basic} + D_i$ (gate, π_{tmp})
        end
        $H_{tentative} \leftarrow g_{tmp}.cost$ (G, D_i, π_{tmp});
        Update the extended layer E;
        $H_{extend} \leftarrow 0$;
        for gate ∈ E **do**
          $H_{extend} \leftarrow H_{extend} + D_i$ (gate, π_{tmp});
        end
        $H \leftarrow \frac{1}{|F+N_{tent}|}(H_{basic} + H_{tentative}) + \frac{W}{|E|} H_{extend}$
      end
      Choose the best gate g_n;
      $\pi_c \leftarrow$ Map_Update (g_n, π_c);
    end
    Update the First layers;
  end
  return schedule
end

REFERENCES

[1] 27-qubit backend: IBM Q team, "IBM Q 27 toronto backend specification V1.0.7," (2020). Retrieved from https://quantum-computing.ibm.com.

[2] 65-qubit backend: IBM Q team, "IBM Q 65 manhattan backend specification V1.0.5," (2020). Retrieved from https://quantum-computing.ibm.com.

[3] Github repository of the hardware-aware multi-programming approach. https://github.com/peachnuts/Multiprogramming.

[4] Ali J Abhari, Arvin Faruque, Mohammad J Dousti, Lukas Svec, Oana Catu, Amlan Chakrabati, Chen-Fu Chiang, Seth Vanderwilt, John Black, and Fred Chong. Scaffold: Quantum programming language. Technical report, Princeton Univ NJ Dept of Computer Science, 2012.

[5] Abdullah Ash-Saki, Mahabubul Alam, and Swaroop Ghosh. Analysis of crosstalk in nisq devices and security implications in multi-programming regime. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 25–30, 2020.

[6] Abdullah Ash-Saki, Mahabubul Alam, and Swaroop Ghosh. Experimental characterization, modeling, and analysis of crosstalk in a quantum computer. *IEEE Transactions on Quantum Engineering*, 2020.

Table 2: Comparison of fidelity when executing two circuits simultaneously on IBM Q 27 Toronto.

Benchmarks		Independent						Correlated											Comparison	
ID		HA			PHA			CDAP			QHSP				GSP				Δ_{PST}%	
ID1	ID2	PST1	PST2	Avg	PST1	PST2	Avg	PST1	PST2	Avg	PST1	PST2	Avg	t	PST1	PST2	Avg	t	Indp.	Corr.
1	1	0.571	0.558	0.565	0.686	0.676	0.681	0.597	0.506	0.552	0.675	0.641	0.658	0.009	0.641	0.682	0.662	0.4	20.6	19.3
1	2	0.334	0.75	0.542	0.661	0.789	0.725	0.522	0.585	0.554	0.69	0.789	0.74	0.012	0.69	0.789	0.74	7.4	33.8	33.6
1	3	0.547	0.412	0.48	0.687	0.591	0.639	0.616	0.487	0.552	0.619	0.552	0.586	0.007	0.619	0.552	0.586	7.4	100	6.2
1	4	0.476	0.45	0.463	0.574	0.642	0.608	0.562	0.158	0.36	0.626	0.647	0.637	0.016	0.626	0.647	0.637	7.4	31.3	76.8
1	5	0.495	0.445	0.47	0.673	0.582	0.628	0.561	0.437	0.499	0.647	0.511	0.579	0.012	0.647	0.511	0.579	1.6	33.5	16
2	2	0.647	0.53	0.589	0.78	0.775	0.778	0.567	0.426	0.5	0.808	0.591	0.7	0.006	0.808	0.591	0.7	14.4	32.1	40.9
2	3	0.428	0.304	0.366	0.787	0.626	0.707	0.635	0.602	0.619	0.764	0.529	0.647	0.013	0.764	0.529	0.647	15	93	4.5
2	4	0.561	0.607	0.584	0.791	0.645	0.718	0.483	0.431	0.457	0.788	0.467	0.628	0.008	0.788	0.467	0.628	14.7	23	37.3
2	5	0.573	0.311	0.442	0.796	0.568	0.682	0.534	0.506	0.52	0.774	0.531	0.653	0.006	0.774	0.531	0.653	8.7	54.3	25.5

Avg: average of PSTs. **t**: runtime in seconds of the partition process. $\Delta_{\mathbf{PST}}$: comparison of average fidelity.

Table 3: Comparison of number of additional gates when executing two circuits simultaneously on IBM Q 27 Toronto.

Benchmarks		Independent						Correlated		Comparison	
ID		HA			PHA			CDAP	QHSP	Δ_g%	
ID1	ID2	g_1	g_2	Sum	g_1	g_2	Sum	g	g	Indp.	Corr.
1	1	12	12	24	12	12	24	42	24	0	42.9
1	2	12	9	21	12	6	18	42	18	14.3	57.1
1	3	12	15	27	12	15	27	57	27	0	52.6
1	4	12	24	36	12	24	36	48	33	0	31.3
1	5	12	18	30	12	18	30	60	30	0	50
2	2	6	12	18	6	6	12	42	15	33.3	64.3
2	3	9	15	24	6	15	21	51	18	12.5	64.7
2	4	9	24	33	6	21	27	54	27	18.2	50
2	5	6	18	24	6	18	24	57	24	0	57.9

g: number of additional gates. **Sum**: sum of number of additional gates. $\Delta_{\mathbf{g}}$: comparison of sum of number of additional gates.

Table 4: Comparison of fidelity when executing three circuits simultaneously on IBM Q 65 Manhattan.

Benchmarks			Independent								Correlated									Comparison	
ID			HA				PHA				CDAP				QHSP					Δ_{PST}%	
ID1	ID2	ID3	PST1	PST2	PST3	Avg	PST1	PST2	PST3	Avg	PST1	PST2	PST3	Avg	PST1	PST2	PST3	Avg	t	Indp.	Corr.
1	2	3	0.61	0.566	0.624	0.6	0.651	0.624	0.555	0.61	0.566	0.57	0.177	0.438	0.609	0.526	0.714	0.616	0.047	1.7	40.8
1	2	4	0.521	0.683	0.289	0.5	0.637	0.703	0.48	0.607	0.163	0.624	0.131	0.306	0.559	0.708	0.531	0.599	0.048	21.9	95.9
1	2	5	0.627	0.725	0.368	0.573	0.623	0.653	0.487	0.588	0.15	0.466	0.233	0.283	0.609	0.592	0.528	0.576	0.047	2.5	103.7
2	3	4	0.644	0.434	0.389	0.489	0.631	0.566	0.544	0.58	0.547	0.156	0.211	0.305	0.633	0.565	0.498	0.565	0.04	18.7	85.6
2	3	5	0.689	0.617	0.488	0.598	0.585	0.542	0.486	0.538	0.548	0.276	0.237	0.354	0.7	0.528	0.34	0.523	0.04	-10	47.8

Avg: average of PSTs. **t**: runtime in seconds of the partition process. $\Delta_{\mathbf{PST}}$: comparison of average fidelity.

Table 5: Comparison of number of additional gates when executing three circuits simultaneously on IBM Q 65 Manhattan.

Benchmarks			Independent								Correlated		Comparison	
ID			HA				PHA				CDAP	QHSP	Δ_g%	
ID1	ID2	ID3	g_1	g_2	g_3	Sum	g_1	g_2	g_3	Sum	g	g	Indp.	Corr.
1	2	3	12	12	12	36	12	6	12	30	75	30	16.7	60
1	2	4	12	9	21	42	12	6	18	36	69	36	14.3	47.8
1	2	5	12	9	18	39	12	6	18	36	78	36	7.7	53.8
2	3	4	9	15	18	42	6	12	18	36	84	39	14.3	53.6
2	3	5	9	15	18	42	9	12	18	39	93	36	7.1	61.3

g: number of additional gates. **Sum**: sum of number of additional gates. $\Delta_{\mathbf{g}}$: comparison of sum of number of additional gates.

[7] Radoslaw C Bialczak, Markus Ansmann, Max Hofheinz, Erik Lucero, Matthew Neeley, AD O'Connell, Daniel Sank, Haohua Wang, James Wenner, Matthias Steffen, et al. Quantum process tomography of a universal entangling gate implemented with josephson phase qubits. *Nature Physics*, 6(6):409–413, 2010.

[8] Carlos Bravo-Prieto, Ryan LaRose, Marco Cerezo, Yigit Subasi, Lukasz Cincio, and Patrick Coles. Variational quantum linear solver: A hybrid algorithm for linear systems. *Bulletin of the American Physical Society*, 65, 2020.

[9] A Robert Calderbank, Eric M Rains, PM Shor, and Neil JA Sloane. Quantum error correction via codes over gf (4). *IEEE Transactions on Information Theory*, 44(4):1369–1387, 1998.

[10] A Robert Calderbank and Peter W Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54(2):1098, 1996.

Table 6: Comparison of fidelity when executing four circuits simultaneously on IBM Q 65 Manhattan.

Benchmarks				Independent										Correlated										Comparison	
ID				HA					PHA					CDAP					QHSP						Δ_{PST}%	
ID1	ID2	ID3	ID4	PST1	PST2	PST3	PST4	Avg	PST1	PST2	PST3	PST4	Avg	PST1	PST2	PST3	PST4	Avg	PST1	PST2	PST3	PST4	Avg	t	Indp.	Corr.
1	2	3	4	0.512	0.622	0.486	0.35	0.493	0.588	0.644	0.572	0.443	0.562	0.145	0.625	0.383	0.283	0.359	0.443	0.747	0.542	0.443	0.544	0.06	14.1	51.5
1	2	3	5	0.44	0.644	0.608	0.203	0.474	0.648	0.638	0.561	0.491	0.585	0.157	0.619	0.511	0.475	0.441	0.612	0.645	0.581	0.373	0.553	0.058	23.4	25.5
1	3	4	5	0.6	0.542	0.228	0.289	0.415	0.592	0.504	0.497	0.404	0.499	0.123	0.608	0.468	0.145	0.336	0.557	0.53	0.32	0.426	0.458	0.058	20.4	36.4
2	3	4	5	0.643	0.544	0.287	0.278	0.438	0.699	0.53	0.525	0.465	0.555	0.271	0.489	0.154	0.138	0.263	0.691	0.477	0.492	0.369	0.507	0.048	26.7	92.9

Avg: average of PSTs. **t**: runtime in seconds of the partition process. Δ_{PST}: comparison of average fidelity.

Table 7: Comparison of number of additional gates when executing three circuits simultaneously on IBM Q 65 Manhattan.

Benchmarks				Independent									Correlated		Comparison	
ID				HA					PHA				CDAP	QHSP	Δ_g%	
ID1	ID2	ID3	ID4	g_1	g_2	g_3	g_4	Sum	g_1	g_2	g_3	g_4	Sum	g	g	Indp.	Corr.
1	2	3	4	12	9	15	24	60	12	9	15	18	54	102	51	10	50
1	2	3	5	12	9	15	12	48	12	6	12	18	48	114	51	0	55.3
1	3	4	5	12	15	18	18	63	12	12	18	18	60	129	60	4.8	53.5
2	3	4	5	6	15	21	18	60	6	15	18	18	57	126	54	5	57.1

g: number of additional gates. **Sum**: sum of number of additional gates. Δ_g: comparison of sum of number of additional gates.

[11] Yudong Cao, Jonathan Romero, Jonathan P Olson, Matthias Degroote, Peter D Johnson, Mária Kieferová, Ian D Kivlichan, Tim Menke, Borja Peropadre, Nicolas PD Sawaya, et al. Quantum chemistry in the age of quantum computing. *Chemical reviews*, 119(19):10856–10915, 2019.

[12] M Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *arXiv preprint arXiv:2012.09265*, 2020.

[13] Ophelia Crawford, Barnaby van Straaten, Daochen Wang, Thomas Parks, Earl Campbell, and Stephen Brierley. Efficient quantum measurement of pauli operators. *arXiv preprint arXiv:1908.06942*, 2019.

[14] Andrew W Cross, Lev S Bishop, John A Smolin, and Jay M Gambetta. Open quantum assembly language. *arXiv preprint arXiv:1707.03429*, 2017.

[15] Poulami Das, Swamit S Tannu, Prashant J Nair, and Moinuddin Qureshi. A case for multi-programming quantum computers. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 291–303, 2019.

[16] Xinglei Dou and Lei Liu. A new qubits mapping mechanism for multi-programming quantum computing. In *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*, pages 349–350, 2020.

[17] Eugene F Dumitrescu, Alex J McCaskey, Gaute Hagen, Gustav R Jansen, Titus D Morris, T Papenbrock, Raphael C Pooser, David Jarvis Dean, and Pavel Lougovski. Cloud quantum computing of an atomic nucleus. *Physical review letters*, 120(21):210501, 2018.

[18] Daniel J Egger, Claudio Gambella, Jakub Marecek, Scott McFaddin, Martin Mevissen, Rudy Raymond, Andrea Simonetto, Stefan Woerner, and Elena Yndurain. Quantum computing for finance: state of the art and future prospects. *arXiv preprint arXiv:2006.14510*, 2020.

[19] Alexander Erhard, Joel J Wallman, Lukas Postler, Michael Meth, Roman Stricker, Esteban A Martinez, Philipp Schindler, Thomas Monz, Joseph Emerson, and Rainer Blatt. Characterizing large-scale quantum computers via cycle benchmarking. *Nature communications*, 10(1):1–7, 2019.

[20] Héctor Abraham et al. Qiskit: An open-source framework for quantum computing. https://qiskit.org/, 2019.

[21] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[22] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3):032324, 2012.

[23] Jay M Gambetta, AD Córcoles, Seth T Merkel, Blake R Johnson, John A Smolin, Jerry M Chow, Colm A Ryan, Chad Rigetti, S Poletto, Thomas A Ohki, et al. Characterization of addressability by simultaneous randomized benchmarking. *Physical review letters*, 109(24):240504, 2012.

[24] Pranav Gokhale, Olivia Angiuli, Yongshan Ding, Kaiwen Gui, Teague Tomesh, Martin Suchara, Margaret Martonosi, and Frederic T Chong. Optimization of simultaneous measurement for variational quantum eigensolver applications. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 379–390. IEEE, 2020.

[25] Alexander S Green, Peter LeFanu Lumsdaine, Neil J Ross, Peter Selinger, and Benoît Valiron. Quipper: a scalable quantum programming language. In *Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation*, pages 333–342, 2013.

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

[27] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[28] Cupjin Huang, Xiaotong Ni, Fang Zhang, Michael Newman, Dawei Ding, Xun Gao, Tenghui Wang, Hui-Hai Zhao, Feng Wu, Gengyan Zhang, et al. Alibaba cloud quantum development platform: Surface code simulations with crosstalk. *arXiv preprint arXiv:2002.08918*, 2020.

[29] Hsin-Yuan Huang, Kishor Bharti, and Patrick Rebentrost. Near-term quantum algorithms for linear systems of equations. *arXiv preprint arXiv:1909.07344*, 2019.

[30] Toshinari Itoko, Rudy Raymond, Takashi Imamichi, and Atsushi Matsuo. Optimization of quantum circuit mapping using gate transformation and commutation. *Integration*, 70:43–50, 2020.

[31] 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):242–246, 2017.

[32] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *Physical Review A*, 101(2):022316, 2020.

[33] Benjamin P Lanyon, James D Whitfield, Geoff G Gillett, Michael E Goggin, Marcelo P Almeida, Ivan Kassal, Jacob D Biamonte, Masoud Mohseni, Ben J Powell, Marco Barbieri, et al. Towards quantum chemistry on a quantum computer. *Nature chemistry*, 2(2):106–111, 2010.

[34] Gushu Li, Yufei Ding, and Yuan Xie. Tackling the qubit mapping problem for nisq-era quantum devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1001–1014, 2019.

[35] Pranav Mundada, Gengyan Zhang, Thomas Hazard, and Andrew Houck. Suppression of qubit crosstalk in a tunable coupling superconducting circuit. *Physical Review Applied*, 12(5):054023, 2019.

[36] 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*, pages 1015–1029, 2019.

[37] Prakash Murali, David C McKay, Margaret Martonosi, and Ali Javadi-Abhari. Software mitigation of crosstalk on noisy intermediate-scale quantum computers. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1001–1016, 2020.

[38] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.

[39] Siyuan Niu, Adrien Suau, Gabriel Staffelbach, and Aida Todri-Sanial. A hardware-aware heuristic for the qubit mapping problem in the nisq era. *arXiv preprint arXiv:2010.03397*, 2020.

[40] 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.

[41] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018.
[42] Timothy J Proctor, Arnaud Carignan-Dugas, Kenneth Rudinger, Erik Nielsen, Robin Blume-Kohout, and Kevin Young. Direct randomized benchmarking for multiqubit devices. *Physical review letters*, 123(3):030503, 2019.
[43] Jonathan Romero and Alan Aspuru-Guzik. Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. *arXiv preprint arXiv:1901.00848*, 2019.
[44] Mohan Sarovar, Timothy Proctor, Kenneth Rudinger, Kevin Young, Erik Nielsen, and Robin Blume-Kohout. Detecting crosstalk errors in quantum information processors. *Quantum*, 4:321, 2020.
[45] Sarah Sheldon, Easwar Magesan, Jerry M Chow, and Jay M Gambetta. Procedure for systematically tuning up cross-talk in the cross-resonance gate. *Physical Review A*, 93(6):060302, 2016.
[46] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
[47] Hao Tang, Anurag Pal, Lu-Feng Qiao, Tian-Yu Wang, Jun Gao, and Xian-Min Jin. Quantum computation for pricing the collateral debt obligations. *arXiv preprint arXiv:2008.04110*, 2020.
[48] 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*, pages 987–999, 2019.
[49] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler. RevLib: An online resource for reversible functions and reversible circuits. In *Int'l Symp. on Multi-Valued Logic*, pages 220–225, 2008. RevLib is available at http://www.revlib.org.
[50] Robert Wille, Lukas Burgholzer, and Alwin Zulehner. Mapping quantum circuits to ibm qx architectures using the minimal number of swap and h operations. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2019.
[51] Feng Zhang, Niladri Gomes, Noah F Berthusen, Peter P Orth, Cai-Zhuang Wang, Kai-Ming Ho, and Yong-Xin Yao. Shallow-circuit variational quantum eigensolver based on symmetry-inspired hilbert space partitioning for quantum chemical calculations. *arXiv preprint arXiv:2006.11213*, 2020.
[52] Peng Zhao, Peng Xu, Dong Lan, Ji Chu, Xinsheng Tan, Haifeng Yu, and Yang Yu. High-contrast z z interaction using superconducting qubits with opposite-sign anharmonicity. *Physical Review Letters*, 125(20):200503, 2020.
[53] Pengcheng Zhu, Zhijin Guan, and Xueyun Cheng. A dynamic look-ahead heuristic for the qubit mapping problem of nisq computers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.
[54] 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*, 38(7):1226–1236, 2018.
[55] 5-qubit backend: IBM Q team, "IBM Q 5 valencia backend specification V1.4.0," (2020). Retrieved from https://quantum-computing.ibm.com.
[56] Frank Arute et. al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510, 10 2019. doi: https://doi.org/10.1038/s41586-019-1666-5.

7 ACKNOWLEDGMENT

This work is funded by the QuantUM Initiative of the Region Occitanie, University of Montpellier and IBM Montpellier. The authors would like to thank Xinglei Dou and Lei Liu for the meaningful discussions and exchanges. The authors are very grateful to Adrien Suau for the helpful suggestions and feedback on an early version of this manuscript. We acknowledge use of the IBM Q for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Q team.

8 AUTHOR CONTRIBUTIONS

S.N and A.T.S contributed equally to this work. A.T.S proposed the problem formalism. S.N implemented the algorithms and wrote the paper. A.T.S revised the paper. Both authors reviewed and discussed the analyses and results of the work.

9 COMPETING INTERESTS

The authors declare no competing interests.

10 ADDITIONAL INFORMATION

Correspondence and requests for materials should be addressed to S.N.