The Journal of Systems and Software 214 (2024) 112085

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

FragQC: An efficient quantum error reduction technique using quantum circuit fragmentation ☆

Saikat Basu [a,b,*], Arnav Das [a,c], Amit Saha [a,d,e], Amlan Chakrabarti [a], Susmita Sur-Kolay [f]

[a] A.K.Choudhury School of Information Technology, University of Calcutta, Kolkata, India
[b] LTIMindtree Ltd., Kolkata, India
[c] Department of Computer Science, St. Xavier's University, Kolkata, India
[d] Institut National de Recherche en Informatique et en Automatique (Inria), Paris, France
[e] École Normale Supérieure (ENS), PSL University, Paris, France
[f] Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India

ARTICLE INFO

Keywords:
Quantum circuit fragmentation
Hybrid quantum systems
Quantum error
Graph partitioning
Genetic algorithm
Circuit cutting
Quantum annealing

ABSTRACT

Quantum computers must meet extremely stringent qualitative and quantitative requirements on their qubits in order to solve real-life problems. Quantum circuit fragmentation techniques divide a large quantum circuit into a number of sub-circuits that can be executed on the smaller noisy quantum hardware available. However, the process of quantum circuit fragmentation involves finding an ideal cut that has exponential time complexity and also the classical post-processing required to reconstruct the output. In this paper, we represent a quantum circuit using a weighted graph and propose a novel classical graph partitioning algorithm for selecting an efficient fragmentation that reduces the entanglement between the sub-circuits along with balancing the estimated error in each sub-circuit. We also demonstrate a comparative study of different classical and quantum approaches to graph partitioning for finding such a cut. We present *FragQC*, a software tool that cuts a quantum circuit into sub-circuits when its error probability exceeds a certain threshold. With this proposed approach, we achieve an increase in fidelity of 13.38% compared to direct execution without cutting the circuit, and 7.88% over the state-of-the-art ILP-based method for the benchmark circuits.

The code for *FragQC* is available at https://github.com/arnavdas88/FragQC.

1. Introduction

Modern researchers have been developing quantum algorithms to achieve asymptotic improvement with the advance of the technology for quantum computing over the past two decades (Nielsen and Chuang, 2010). Through qubits and quantum gates, a quantum algorithm may be implemented as a quantum circuit. Despite the enormous potential of quantum algorithms, the current generation of quantum computers is error-prone, which restricts their capacity to solve computational problems (Preskill, 2018).

The use of quantum error correcting codes (QECCs) (Grassl et al., 1999; Gottesman, 1999; Baek et al., 2019; Leuenberger and Loss, 2001; Shor, 1995; Steane, 1996; Wootton, 2020; Terhal, 2015; Laflamme et al., 1996) to eliminate noise-related errors opens the door to fault-tolerant quantum computation (Campbell et al., 2017). However, in reality, implementing quantum error correction entails a significant cost due to the high number of qubit requirements, which is still outside the scope of current near-term devices. Numerous error reduction approaches have been developed as a result of the typical error rate of present near-term technologies; one of them is quantum error mitigation (QEM) (Nautrup et al., 2019; Kim et al., 2020). QEM does not use additional quantum resources; instead, it uses a variety of techniques, including extrapolation, probabilistic error cancellation, quantum subspace expansion, symmetry verification, machine learning, etc.,Cai et al. (2023) to improve the accuracy of estimating the outcome in a particular quantum computational problem slightly. According to the most recent research, QEM is constrained to quantum circuits with a small number of qubits and a small depth because of the significant overhead of classical computational time complexity (Nautrup et al., 2019).

☆ Editor: Prof Raffaela Mirandola.

* Corresponding author at: LTIMindtree Ltd., Kolkata, India.
E-mail addresses: saikat.basu@ltimindtree.com (S. Basu), arnav.das88@gmail.com (A. Das), amit.saha@inria.fr (A. Saha), acakcs@caluniv.ac.in (A. Chakrabarti), ssk@isical.ac.in (S. Sur-Kolay).

https://doi.org/10.1016/j.jss.2024.112085
Received 30 September 2023; Received in revised form 14 March 2024; Accepted 25 April 2024
Available online 30 April 2024

Fragmentation of quantum circuits (Bravyi et al., 2016; Peng et al., 2020; Tang and Martonosi, 2022) can be a helpful strategy for overcoming the technical difficulties of QEM since it partitions a quantum circuit into smaller sub-circuits, with fewer qubits and smaller depth. The short coherence periods of noisy intermediate-scale quantum (NISQ) processors must thus be handled by the sub-circuits. When running on a NISQ device, each sub-circuit experiences less noise. On a small quantum computer, a larger quantum system is primarily simulated through the fragmentation of a quantum circuit. In Tang et al. (2021) and Ayral et al. (2020), the authors suggested fragmenting a quantum circuit to reduce the exponential post-processing cost. In *CutQC* (Tang et al., 2021), the authors showed enhanced fidelity, which is an addition to the noisy compiler that is already available. Moreover, earlier compilers would not support the execution of circuits with sizes larger than that of available quantum computers. Therefore, *CutQC* is state-of-the-art because it can be used with any compiler to execute circuits that are larger in size and yet attain higher fidelity. Researchers also investigated for the first time in Ayral et al. (2021) how such fragmentation impacts the various quantum noise models. Circuit fragmentation was later examined in Perlin et al. (2021a) as a way to reduce the impacts of noise. Although the main goal of the earlier research was to break up complex circuits so that they could be implemented, the issue of noise that might cause false results was never appropriately addressed.

Motivation: The fragmentation of a quantum circuit into smaller sub-circuits can effectively mitigate the adverse effects of noise; however, it comes at the expense of increased post-processing and overall computational costs. A notable observation is that many existing quantum circuit fragmentation techniques do not take into account the presence of noise when determining optimal cuts for a quantum circuit. In the context of addressing this drawback, the authors of Basu et al. (2022) designed *i-QER*, a novel quantum circuit fragmentation approach based on machine learning to predict errors. Notably, this is the first tool in the literature, to the best of our knowledge, where errors in a quantum circuit are explicitly taken into consideration during the quantum circuit fragmentation process. However, the major limitations of *i-QER* are: (i) the machine learning-based approach has a scalability issue, especially with accurate training of the model when the circuit size is large, (ii) it is always a hardware-dependent method while using a machine learning approach, and (iii) the choice of an appropriate machine learning model from a plethora of those is still an open question.

This paper aims to address all the above-mentioned issues by proposing a generalized circuit fragmentation approach by optimizing both the success probability and the cut size in a graph-algorithmic manner instead of machine learning.

Contributions:

- We have mapped a quantum circuit into a node and edge weighted graph by considering both the noise and the two-qubit gate operations.
- This has led us to achieve better fidelity for a quantum circuit by using graph partitioning approaches as compared to the state-of-the-art.
- We have carried out a numerical analysis on benchmark circuits with well-known approaches such as genetic algorithm, *h-Metis*, *Mt-KaHyper*, and quantum annealing as a starting point for our *FragQC* tool to find the most efficient cut among all the approaches used.

In the rest of this paper, Section 2 briefly presents the preliminary concepts of quantum circuit fragmentation, graph partitioning algorithms, and available quantum hardware. Section 3 describes *FragQC*, the proposed tool. Section 4 briefly discusses the experimental results of the proposed methodology with our concluding remarks in Section 5.

2. Background

This section briefly explains quantum circuit fragmentation and its challenges followed by a few relevant existing heuristic and approximate approaches for solving graph partitioning problems. Lastly, we shed some light on existing quantum hardware and their error rates.

2.1. Quantum circuit fragmentation

A theoretical overview of quantum circuit fragmentation and an illustrative example are presented.

2.1.1. The idea

Each line for a qubit of a quantum circuit represents a sequence of single and multi-qubit gate operations. The time flows from left to right in the quantum circuit diagram. Quantum Circuit fragmentation cuts these notional qubit wires vertically.

In Peng et al. (2020), the authors demonstrated mathematically that the idea to cut a qubit wire is based on the notion that, if we have multiple copies of an experimentally generated single qubit state with a density matrix ρ, then the set $\{I/\sqrt{2}, X/\sqrt{2}, Y/\sqrt{2}, Z/\sqrt{2}\}$ forms an orthonormal set of matrices concerning the Hilbert–Schmidt inner product. So ρ may be expanded as:

$$\rho = \frac{Tr(\rho)I + Tr(\rho X)X + Tr(\rho Y)Y + Tr(\rho Z)Z}{2}. \tag{1}$$

In order to run on quantum computers, the Pauli matrices can be further decomposed into their eigenbases (Tang et al., 2021) as follows:

$$\rho = \frac{\rho_1 + \rho_2 + \rho_3 + \rho_4}{2} \tag{2}$$

where

$$\begin{aligned}
\rho_1 &= [Tr(\rho I) + Tr(\rho Z)]|0\rangle\langle 0| \\
\rho_2 &= [Tr(\rho I) - Tr(\rho Z)]|1\rangle\langle 1| \\
\rho_3 &= Tr(\rho X)[2|+\rangle\langle +| - |0\rangle\langle 0| - |1\rangle\langle 1|] \\
\rho_4 &= Tr(\rho Y)[2|+i\rangle\langle +i| - |0\rangle\langle 0| - |1\rangle\langle 1|]
\end{aligned}$$

Physically, the trace operators are equivalent to measuring the qubits in one of the Pauli bases $\sigma_i \in \{I, X, Y, Z\}$, and the density matrices correspond to physically initializing the qubits to one of the eigenstates.

If we assume that a qubit wire connecting A and B, two vertices representing gates, is cut, then we can reconstruct the sub-circuits by summing over the four pairs of measurement circuits added to A and an initialization circuit added to B. After that, the overall circuit output is reconstructed by adding the four pairs of Kronecker products between the sub-circuit outputs. As a result, there are 4^k Kronecker products to be computed for a cut size of k (Peng et al., 2020).

Example of quantum circuit fragmentation. Let us consider a quantum circuit with five qubits and four two-qubit *CNOT* gates shown in Fig. 1(a). All the qubits are initialized to $|0\rangle$. In order to implement quantum circuit fragmentation, first we need to construct a graph from the given circuit, where the vertices are the two-qubit gates and there is an edge if two two-qubit gates have at least one in common. Thus, for a given quantum circuit C, we have a graph $G(V, E)$. The task is to find a cut that can separate the vertices into more than one disjoint set as shown in Fig. 1(b).

The circuit C can be partitioned into two sub-circuits, as shown in Fig. 1(b). Here, the two partitions $\{A, B\}$ & $\{C, D\}$ are separated by the dashed arrow line. The number of edges between the two partitions can be defined as the cut size (k). Considering that only one qubit, i.e., the third qubit (q[2]) is connecting the partitions, the value of k is 1. Therefore, each sub-circuit can be executed on a 3-qubit quantum hardware instead of a 5-qubit one. We need to take measurements on the 3rd-qubit after the *CNOT* (node B). The initialization of the

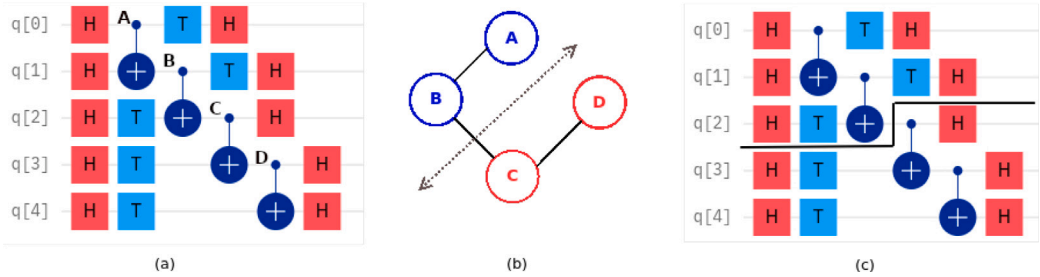


Fig. 1. Example of quantum circuit fragmentation: (a) a quantum circuit C with 5 qubits and 4 two-qubit gates; (b) the corresponding graph G of C; (c) the two quantum sub-circuits after a fragmentation.

sub-circuit $\{C, D\}$ has to be done based on the measurements after B. Therefore, conventionally in the classical post-processing step, the complete probability distribution for the entire circuit can be reconstructed by taking the corresponding outputs of the two smaller sub-circuits, running four pairs of Kronecker products, and adding them together.

In Tang et al. (2021) and Tang and Martonosi (2022), the authors have demonstrated efficient ways for the classical reconstruction method. In Perlin et al. (2021b), the authors have proposed maximum-likelihood fragment tomography (MLFT) as an improved circuit fragmentation technique, with a limited number of qubits to run the quantum sub-circuits on quantum hardware. MLFT further finds the most likely probability distribution for the output of a quantum circuit with the measurement data obtained from the circuit's fragments, along with minimizing the classical computing overhead of circuit fragmentation methods. Hence, they showed that circuit fragmentation as a standard tool can be used for running the sub-circuits on quantum devices by estimating the outcome of a partitioned circuit with higher fidelity as compared to the full circuit execution.

2.1.2. Challenges of quantum circuit fragmentation

In spite of this immense potential, quantum circuit fragmentation faces a few formidable challenges when it is applied to large quantum circuits. Finding an efficient cut location is the first difficult task. Quantum circuits can always be divided into smaller sub-circuits, but choosing an efficient cut is critical for reducing the amount of classical post-processing and the effects of noise. Partitioning a large quantum circuit into sub-circuits often requires multiple edges or qubit wires to be cut. In such cases, all the possible measurement and initialization combinations have to be evaluated. Hence, the number of Kronecker products required is 4^k, with k being the number of edges cut. Thus, quantum circuits with n edges have a combinatorially explosive search space of $\mathcal{O}(n!)$ to find an efficient cut. Cutting a quantum circuit is expensive in terms of classical post-processing costs; thus, we should cut a quantum circuit if and only if it is necessary. Therefore, to improve the fidelity of the quantum circuit, we should only consider bi-partitioning the circuit and check whether we can achieve reasonable fidelity (Basu et al., 2022).

Additionally, if we consider the effects of noise and look to improve the fidelity of a quantum circuit using the quantum circuit fragmentation approach, the problem of finding an efficient cut becomes even more complex. Section 3.1 addresses this problem with different classical as well as quantum annealing-based approaches. Before that, let us describe a few popular graph partitioning algorithms very briefly.

2.2. Graph partitioning algorithms

In this article, we focus on the problem of balanced bi-partitioning of a graph for quantum circuit fragmentation. The goal is to partition the graph into two subgraphs with nearly equal disjoint sets of vertices while minimizing the capacity of the edges between the two subgraphs. The problem being NP-complete, greedy heuristics such as Kernighan–Lin (KL) algorithm (Kernighan and Lin, 1970) and Fiduccia–Mattheyses (FM) algorithm (Fiduccia and Mattheyses, 1982) came into vogue. Then, *h-METIS* and Multi-Threaded Karlsruhe Hypergraph Partitioner (Mt-KaHyPar) emerged as the two most popular hypergraph partitioning methods, which we describe briefly below. Further, we also describe a genetic algorithm and a quantum annealing-based method.

h-METIS. Proposed by Karypis and Kumar (1998a,b), it partitions large hypergraphs, mostly generated while circuit design. It is based on multilevel graph partitioning, as explained in Karypis and Kumar (1999) and Karypis et al. (1997). Unlike other graph partitioning algorithms, *h-METIS* does not perform partitioning operations on the original graph. It takes a coarsening approach repeatedly, where the vertices and edges of the given graph are collapsed to reduce it to a smaller graph. It then performs the partitioning operation on the small graph. In the next phase, it performs an uncoarsening on the two subgraphs obtained along with refinements to the partitioning. In this manner, *h-METIS* can quickly produce high-quality partitions for a large variety of hypergraphs. Experimental results in Karypis and Kumar (1998b) on a large number of hypergraphs show that *h-METIS* consistently produces better partitions than those by other widely used algorithms, such as KL, and FM. We leverage this consistent tool for circuit partitioning to compare with our approach in our experiments on *FragQC*.

Mt-KaHyPar. This shared-memory multilevel graph and hypergraph partitioner is equipped with parallel implementations of techniques used in the best sequential partitioning algorithms. It can partition extremely large hypergraphs very fast and with high quality. It performs balanced partitioning while minimizing an objective function defined on the edges. It can optimize the cut-net, connectivity, sum-of-external-degree, and Steiner tree metric. However, it is to be noted that the quantum circuit fragmentation brings additional exponential post-processing costs. Thus, we have restricted our quantum circuit fragmentation to only bi-partition in our proposed technique. Hence, the suggested objective functions like sum-of-external-degree Metric and Steiner tree metric are automatically reduced to the cut-net metric.

The cut-net metric is defined as the total weight of all nets spanning more than one block of the graph partition Π (also called cut nets) and can be expressed as follows:

$$f_c(\Pi) := \sum_{e \in E_{\text{Cut}}(\Pi)} \omega(e).$$

Genetic algorithm. Genetic algorithm (GA) is a metaheuristic inspired by natural selection. GAs, which rely on biologically inspired operators such as selection, crossover, and mutation, are often employed to find near-optimal solutions to optimization and search problems. It starts with an initial set of solutions called an initial population, which evolves into different populations with each iteration or generation. After multiple generations, the algorithm returns the best member of the population as the solution to the given problem.

In each generation, two members of the population are chosen and are then combined to create offspring by using a crossover operator. The mutation operator further modifies an offspring with a very low probability of including more diversity in the population.

In Bui and Moon (1996), the authors have applied a GA for the graph bi-partitioning problem and have reported performance comparable to or better than KL, FM, and simulated annealing based algorithms. Thus, we plan to use the basic idea of the GA to construct our classical approach for solving the graph bi-partitioning problem.

Quantum annealing. The Hamiltonian of a physical system represents its total energy. If the Hamiltonian of a system evolved very slowly from an initial state to a final state, then the adiabatic theorem (Born and Fock, 1928) states that, if the system is in the nth eigenstate of the initial Hamiltonian, it evolves as the nth eigenstate of the final Hamiltonian.

In Farhi et al. (2000), the authors proposed a quantum annealing based algorithm that leverages this adiabatic evolution theorem to solve different combinatorial optimization problems. The initial state of the system is chosen to be the ground state of a simple Hamiltonian, which is then slowly evolved into the final Hamiltonian, whose ground state encodes the solution to the problem. Therefore, if the evolution is slow enough, then according to the adiabatic theorem, the system remains in its ground state throughout the evolution, and we have the solution to our problem in the final state.

If we can encode the objective function of a graph bi-partitioning problem into a Hamiltonian, then the QA can find the optimal partitioning strategy. In this paper, the graph partitioning algorithm is used to balance the error of a quantum circuit into two quantum subcircuits. Thus, before taking a look into the proposed methodology, it is important to discuss the quantum hardware and its error rates.

2.3. Quantum hardware and its error rate

Superconducting quantum devices (Koch et al., 2007), quantum dots (Loss and DiVincenzo, 1998), ion traps (Bruzewicz et al., 2019), and neutral atoms (Graham et al., 2022) are currently the most popular quantum technologies for constructing qubits and quantum gates. For hardware compatibility, the quantum logic gates in a quantum circuit, such as *CNOT, Hadamard, S, T, X, Y, Z*, must be decomposed using primitive gate operations of a specific quantum hardware or NISQ (Noisy Inter-mediate Scale Quantum) (Preskill, 2018) device. Furthermore, each quantum device has a dedicated qubit connectivity topology, since each is built with its own unique set of physical qubits, coupling strengths, and control mechanisms. These variations can result in different noise characteristics. Some devices, for example, may contain qubits that are closer to one another, resulting in stronger interactions and perhaps greater crosstalk between qubits. Thus, quantum hardware has hardware-specific properties such as gate error rates, readout errors, etc. In this paper, we consider IBM's superconducting-based hardware for further experiments. Fig. 2 depicts the error map of *ibm_nairobi*. It shows the qubit connectivity layout as well as the error rates for different gate operations in the qubits.

Along with the error rates of different gate operations, there are a few critical hardware-specific properties, such as relaxation time, and coherence time. The relaxation time T_1 is a crucial parameter that gives the period during which a qubit can remain in a superposition state or maintain its quantum information before decohering into a classical state. The coherence time T_2 represents the duration during which a quantum system can preserve the quantum phase information that enables quantum computations.

While there are a handful of quantum devices available to us for performing our experiments, these can be performed on any other available quantum hardware with a sufficient number of qubits so that we can easily validate our tool on medium and large-scale quantum circuits.

3. Our proposed tool: FragQC

An overview of *FragQC*, a tool for reducing quantum errors, is provided below. Its flowchart with the essential modules is given in Fig. 3.

The tool accepts a quantum circuit and hardware information as input and calculates the potential success probability of the quantum circuit while considering the noise profile of a specific hardware. A novel error-influenced balanced bi-partitioning algorithm is executed to partition the circuit into two sub-circuits whenever the projected success probability is below a certain user-specified threshold. This bi-partitioning is continued recursively until the sub-circuits can be run with a reasonable chance of success. For the sake of simplicity, we consider the success probability threshold of each sub-circuit akin to the success probability threshold of the overall circuit, which is given by the user. After the sub-circuits are implemented on the hardware, their outputs (probability distributions) are combined appropriately to generate the output of the entire circuit. We have adopted the output reconstruction method from Tang et al. (2021).

3.1. Proposed technique on circuit fragmentation

Here we elaborate upon the proposed method of circuit fragmentation, as shown in Fig. 4.

3.1.1. Graph representation of a quantum circuit

First, we represent the given quantum circuit C as a graph G_C. We denote each two-qubit gate in C as a vertex, and an edge between two vertices indicates that the corresponding two-qubit gates share one or two qubits. The weight of the edge is either 1 or 2 depending on the number of qubits shared by the two two-qubit gates.

Our goal is to cut the quantum circuit in such a way that it not only decreases the interaction between the two partitions but also balances the probability of error in each partition. When executed on quantum hardware separately, this reduces the impact of noise. In order to ensure this, we intend to store the error probability information of the circuit for specific quantum hardware as the vertex weight of the graph. Before the details of the calculation of the vertex weight from the error probabilities are given, we briefly discuss the quantum error model.

Quantum error model. Quantum errors can be broadly categorized into two major types: errors due to noisy gate operation and errors due to idle qubits. The noise model can be expressed in terms of the Kraus operators (Nielsen and Chuang, 2011). Let us consider a pure state ψ, and its density matrix $\sigma = |\psi\rangle\langle\psi|$. The evolution of the state ψ in a quantum channel can be given by a function ξ of its density matrix σgiven as

$$\xi(\sigma) = \sum_i K_i \sigma K_i^\dagger, \tag{3}$$

where K_i is the Kraus operator and $K_i^\dagger$ is complex conjugate transpose of K_i. The evolution of a noisy quantum system can also be represented by Eq. (3) (Nielsen and Chuang, 2010). If we consider depolarizing noise channels, then the Kraus operators are the Pauli matrices.

1. Gate operation error: The possible error model for a quantum system with one qubit can be expressed as:

$$\xi(\sigma) = \sum_{i\in\{0,1\}} \sum_{j\in\{0,1\}} p_{i,j} (X^i Z^j) \sigma (X^i Z^j)^\dagger. \tag{4}$$

where X and Z are Pauli operators, and the probability of the corresponding Kraus operator is denoted by $p_{i,j}$.

Hence, the possible quantum error channels are

if $i = 0$ and $j = 0$ then $X^0 Z^0 = I$, i.e., no error
if $i = 0$ and $j = 1$ then $X^0 Z^1 = Z$, i.e., phase flip error
if $i = 1$ and $j = 0$ then $X^1 Z^0 = X$, i.e., bit flip error

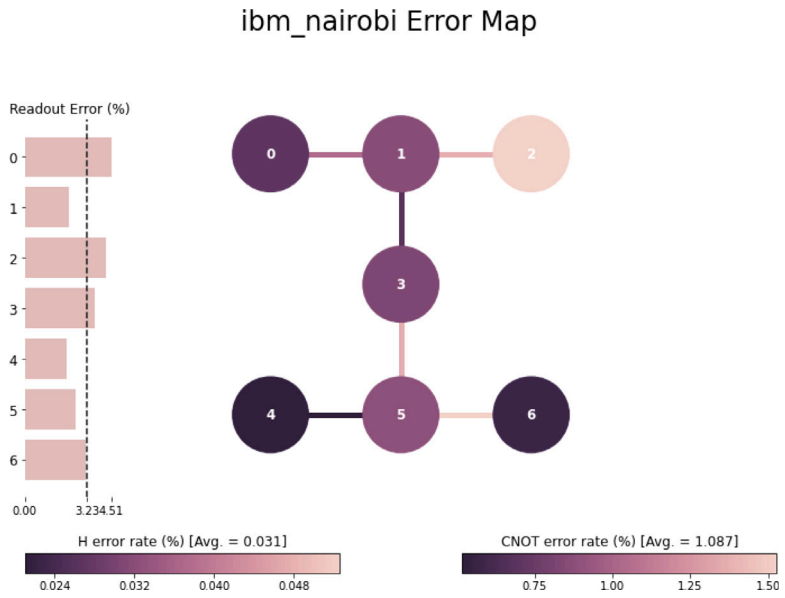


Fig. 2. Error map of 7-qubit *ibm_nairobi* device.

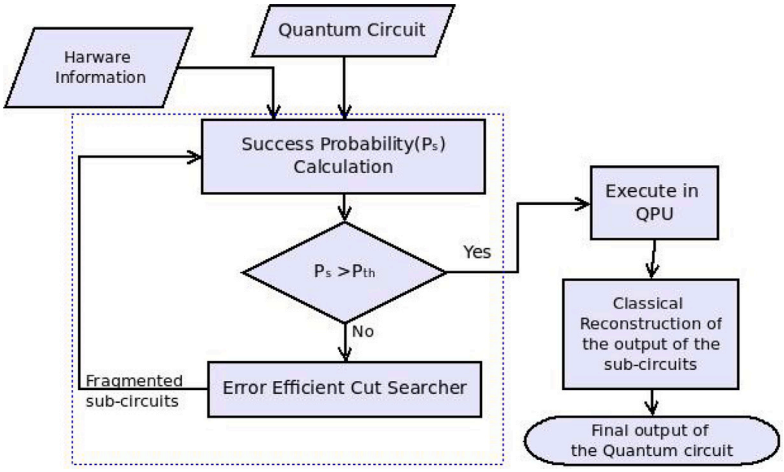


Fig. 3. Flowchart of the proposed tool *FragQC*.

if $i = 1$ and $j = 1$ then $X^1 Z^1 = XZ$, i.e., both bit and phase flip errors

In Fowler et al. (2012), the authors represented a noisy gate operation by the ideal gate operation followed by a set of Pauli operators $\{X, Y, Z\}$ with probability p_{ex}, p_{ey}, and p_{ez} respectively.

2. Amplitude damping error: When a qubit in an open quantum system is kept idle, it can absorb or dissipate energy and change its state spontaneously over time. Let us assume that a qubit can dissipate and absorb energy with a probability of p and $(1 - p)$ respectively. This noise channel is called an amplitude damping channel, and it can also be defined using Kraus operators. The Kraus operators for energy dissipation or state change $|1\rangle \rightarrow |0\rangle$ are written as:

$$K_0 = \sqrt{p}\begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\lambda} \end{bmatrix}, \quad K_1 = \sqrt{p}\begin{bmatrix} 0 & 0 \\ 0 & \sqrt{\lambda} \end{bmatrix},$$

The Kraus operators for energy absorption or state change $|0\rangle \rightarrow |1\rangle$ are written as Nielsen and Chuang (2010):

$$K_2 = \sqrt{1-p}\begin{bmatrix} \sqrt{1-\lambda} & 0 \\ 0 & 1 \end{bmatrix} \;\&\; K_3 = \sqrt{1-p}\begin{bmatrix} 0 & 0 \\ \sqrt{\lambda} & 0 \end{bmatrix}.$$

Here, $\lambda \propto e^{-\tau/T_1}$, where T_1 is called the energy relaxation time of the quantum system and τ is the time duration of the quantum system or the time duration for which the quantum circuit is operational.

3. Phase damping error: Phase damping is a unique quantum mechanical noise model that describes the loss of quantum coherence without loss of energy. The Kraus operators for the dephasing channel can be given as

$$K_{p0} = \sqrt{p}\begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\lambda} \end{bmatrix}, \quad K_{p1} = \sqrt{p}\begin{bmatrix} 0 & 0 \\ 0 & \sqrt{\lambda} \end{bmatrix}.$$

Here, $\lambda \propto e^{-\tau/T_2}$, where T_2 is called the coherence time of the quantum system and τ denotes the time duration.

Computing the vertex weights. In Majumdar et al. (2016), the authors have given a linear time algorithm to trace error in a quantum circuit, and in Majumdar et al. (2021) and Saha et al. (2022), the authors have computed the probability of success for a quantum circuit considering errors due to noisy gate operations and amplitude damping. Inspired by these works, we first compute the error probability of a quantum circuit C. We consider the most common error model for quantum systems, namely errors due to gate error and idle error (Gokhale et al., 2019) modeled with amplitude and phase damping error. Let us assume that C

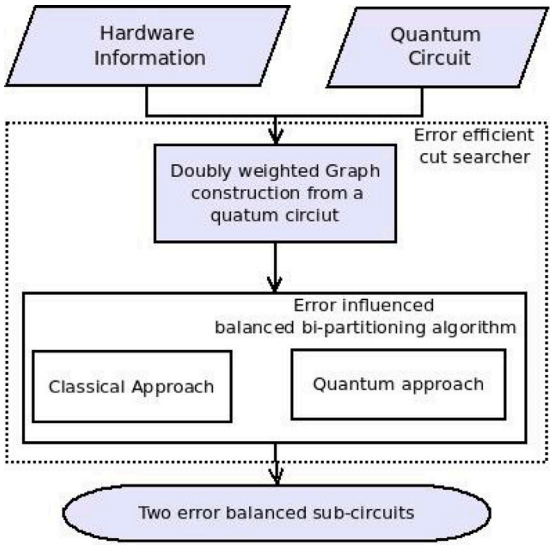


Fig. 4. Block diagram of our error efficient cut searcher. It has two main parts, namely (i) constructing a doubly weighted graph for a given quantum circuit, and (ii) error influenced balanced partitioning algorithm for circuit fragmentation.

has k_1 single-qubit gates, k_2 two-qubit gates, and the error probability of a single-qubit gate and a two-qubit gate is p_1 and p_2 respectively. The error probability of the circuit due to the noisy gate operations can be written as

$$p_{GE} = 1 - \{(1-p_1)^{k_1}(1-p_2)^{k_2}\}. \tag{5}$$

The error due to the amplitude damping and phase damping along with the gate error can therefore be expressed as

$$P_{Error} = 1 - \{(1-p_1)^{k_1}(1-p_2)^{k_2} exp^{-(\tau/T_1+\tau/T_2)}\}. \tag{6}$$

The weight of a vertex from Eq. (6) is the error of the corresponding two-qubit gate, and the product of the errors of the sequence of single-qubit gates operating on the qubit responsible for the edge. We also consider the idle error for those edges, i.e., decoherence prior to that two-qubit gate. We normalize the calculated error probability to improve accuracy and assign it to the vertex as weight. This is to facilitate consistent scaling and smoother convergence.

Example. Let us consider a hardware or backend, with the following error rates. The error rate for the single-qubit and two-qubit gates is 0.00246 and 0.03130 respectively. The time to execute a single-qubit gate and a two-qubit gate is 0.0355 ns and 0.46 ns respectively. The coherence time (T_1) is set to 122.22 μs and the relaxation time (T_2) to 187.75 μs. These are used to generate the adjacency matrix of the quantum circuit. The adjacency matrix is a 2D square $n \times n$ matrix where n is the number of CX gates. The diagonals of this adjacency matrix are replaced by the cumulative error of single-qubit gates in the previous fragment.

Let us illustrate with a quantum circuit C having 8 qubits and 14 two-qubit *CNOT* (CX) gates, as shown in Fig. 5(a). In the corresponding doubly-weighted graph G_C, each vertex represents a *CNOT* (CX) gate and each edge denotes a qubit connecting two *CNOT* gate. Let us calculate the weight of the vertex $CX13$ which has an edge from $CX11$ and from $CX12$. Hence, while computing the vertex weight, we have to consider:

1. the error rate of $CX13$,
2. error due to all the single qubit gates applied between $CX11$ to $CX13$,
3. error for all the single qubit gates operated between $CX12$ to $CX13$
4. amplitude and phase damping error.

Thus the weight of $CX13$ is 0.070 as shown in Fig. 5(b). Similarly, we compute the weights for all the vertices as portrayed in Fig. 5(b).

3.1.2. Error influenced balanced bi-partitioning of circuit

First, we present the objective function of the optimization problem we intend to solve.

Objective function. Let us assume that we have a cut c, which partitions graph G_C into two sub-graphs G_1 and G_2. Let an indicator variable y_v be associated with each vertex $v \in V$ such that,

$$y_v = \begin{cases} 0 & if \quad v \in G_1 \\ 1 & if \quad v \in G_2 \end{cases} \tag{7}$$

If $e_{i,j}$ is the edge connecting the vertices v_i and v_j having edge weight $w_{i,j}$, then the cut size (K_c) for the specific cut c, can be written as

$$K_c = \sum_{e_{i,j} \in E} w_{i,j}(y_{v_i} - y_{v_j})^2. \tag{8}$$

Let the sum of vertex weights for the partitions G_1 and G_2 be Ω_{G_1} and Ω_{G_2} respectively. Hence, the overall cost for a cut c can be written as

$$Cost_c = K_c\left(\frac{1}{\Omega_{G_1}} + \frac{1}{\Omega_{G_2}}\right). \tag{9}$$

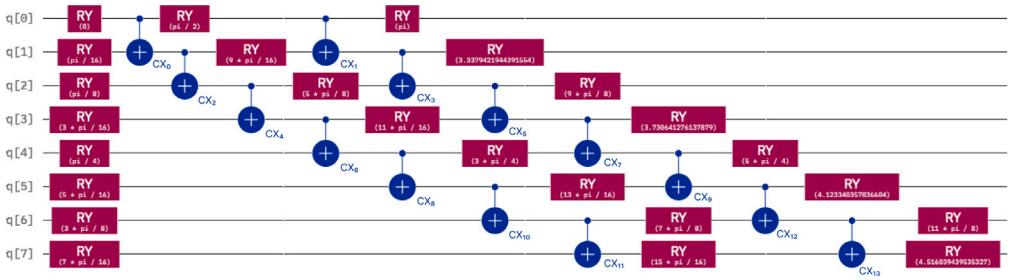
Our aim is to find a cut c for which this $Cost_c$ is minimum. Thus, the cost is given by Eq. (9) is our objective function (Deb et al., 2016) that has to be minimized. In this paper, we apply both classical and quantum approaches to solve this optimization problem to establish the most suitable method. We start with a genetic algorithm-inspired proposed classical approach, followed by a quantum annealing-based approach.

Proposed classical approach for finding an error-balanced min-cut. We propose a GA-inspired greedy heuristic to minimize our objective function (Eq. (9)). Algorithm 1 outlines this approach for identifying a min-cut c in a graph G_C while maintaining balance with respect to errors.

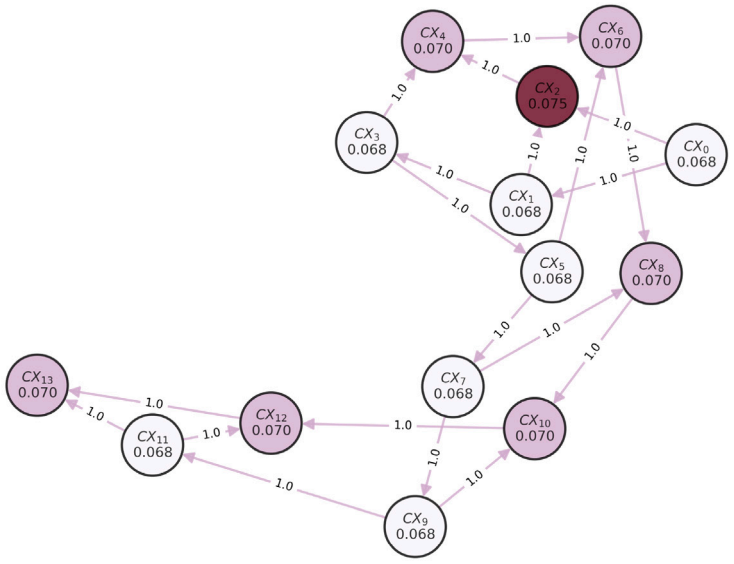
Algorithm 1 Finding an Error-balanced Min-Cut

Input: N, $initialPartitionVector$
Output: $MinPartitionVector$, $MinCost$ ▷ N = number of vertices
  $MinVector = initialPartitionVector + 1$
  $MinCost = CostCalculator(initialPartitionVector)$
  $CostFlag = 0$
  while $CostFlag \leq c_2$ **do**
    for $i \leftarrow 1$ to N **do**
      $partitionVector[i] = initialPartitionVector[i] \oplus 1$
      $cost[i] = CostCalculator(partitionVector[i])$
      if $cost[i] < MinCost$ **then**
        $MinCost = cost[i]$
        $MinVector = partitionVector[i]$
        $costFlag = 0$
      else
        $CostFlag = CostFlag + 1$
      end if
    end for
    $initialPartitionVector = crossover(MinVector, initialPartitionVector)$
  end while
    return $MinCost$, $MinVector$

We initiate the algorithm with a random cut effectively dividing the graph into two sub-graphs. We obtain a partition vector, essentially a string containing the values of y_v for all N vertices of G_C. The cost of



(a) A quantum circuit C.



(b) The doubly-weighted graph G_C with vertex and edge weights (a vertex with higher weight has darker color).

Fig. 5. An example circuit and its corresponding doubly-weighted graph.

this initial partition is computed using Eq. (9). We have provided an Algorithm 3 to compute cut-size, further the Algorithm 3 is leveraged by Algorithm 2, in implementing the cost function in Eq. (9). Detailed explanations of these algorithms are provided in the Appendix A.

The subsequent phases of the process iteratively flip each bit in the partition vector and update $MinCost$ if the cost of the new partition vector is lower. This process is repeated until all bits have been flipped. This marks the completion of one iteration or pass through Algorithm 1. Once a pass is completed, Algorithm 1 prepares the initial partition vector for the next iteration from the partition vector with the lowest cost given by $MinCost$ and the initial partition vector, then applies a single point crossover operation. That is a random crossover point is selected and the tails of the two partition vectors are swapped to get a new partition vector. In Appendix B, we have shown Algorithm 4 to implement the single point crossover function to generate a new partition vector for the next pass.

In summary, the algorithm calculates partition costs, iteratively improves the partition by flipping bits, and uses a crossover operation to create a new initial partition vector for the next iteration, all with the aim of optimizing the partition. This process is repeated until no further improvement in cost is observed. After reaching this point, the algorithm undergoes a few more iterations (a total of c_2 times) before ultimately returning the final $MinCost$ and the corresponding $MinVector$. The overall time complexity of Algorithm 1 is $O(m \cdot n^2)$, where n is the number of vertices and m the total number of iterations.

Quantum annealing based approach. We incorporate quantum annealing as a quantum approach in *FragQC* to solve the proposed optimization problem defined in Eq. (9). In Ushijima-Mwesigwa et al. (2017), the authors have exhibited the graph partitioning using quantum annealing on the D-Wave system. We also use the D-Wave system for our experiments. For minimizing our objective function in Eq. (9) on a D-wave system, we need to describe our objective function in the following Ising objective function

$$\min\left(\sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j\right). \tag{10}$$

where $s_i \in (+1, -1)$ are subject to local fields h_i and pair wise interactions with coupling strengths J_{ij}.

The quadratic unconstrained binary optimization (QUBO) representation is often preferred with its 0/1-valued variables over the Ising $-1/+1$-valued variables because it is more natural. The QUBO objective function is

$$\min\left(\sum_i Q_{ii} x_i + \sum_{i<j} Q_{ij} x_i x_j\right). \tag{11}$$

where Q_{ii} is analogous to the Ising h_i, as are Q_{ij} and J_{ij}. The Ising and QUBO models are related through the transformation $s = 2x - 1$.

Fortunately, the D-Wave machine allows both the Ising and QUBO forms, hence we describe our objective function as Ising formulation.

Our Ising formulation for proposed doubly weighted graph balanced bi-partitioning can be derived from Eq. (9), as follows:

$$\min\left(\sum_{i,j}^{n} w_{ij}^2 \frac{1-s_i s_j}{2} + 2\sum_{i<j}^{n} v_i v_j s_i s_j + \sum_{i=1}^{n} {v_i}^2 + \left(\sum_{i=1}^{n} v_i - \frac{n}{2}\right)^2\right). \quad (12)$$

where w_{ij} is the weight of an edge between vertex i and j and v_i is the weight of a vertex i in G_C. The total number of vertices of the graph is n. The first sum of the function makes sure that the number of edges between two partitions is minimized (Lucas, 2014). The next two sums are used to separate the vertices into two partitions in such a way that the weight of the vertices gets balanced in two partitions. The last term is used to normalize the objective function. We minimize this objective function through quantum annealing on a D-Wave system to find an optimal cut in a quantum circuit.

4. Experimental evaluation of FragQC

We present here the notable results obtained when we fed different benchmark quantum circuits from Quetschlich et al. (2023) and Li et al. (2022) to our tool *FragQC*. We have applied both classical and quantum approaches, i.e., the GA-inspired approach, *Mt-KaHyPar* (classical), and a quantum annealing-based approach implemented using D-Wave's quantum annealing device. We have compared both results with *h-METIS* which is a very popular and state-of-the-art tool for circuit partitioning.

At first, we evaluate different graph partitioning algorithms in terms of solution quality and time-to-solution. Present-day quantum circuits are characterized by a smaller size, resulting in correspondingly modest graphs with a maximum node count typically in the range of a few hundred. Consequently, the time-to-solution for the majority of graph partitioning approaches remains reasonably small. Hence, quality-of-solution takes priority over time-to-solution.

Next, we execute our method on multiple benchmark circuits using our proposed tool, *FragQC*, wherein we estimate the fidelity in comparison to an ideal simulation. A comprehensive study was undertaken to compare the fidelity outcomes of these benchmark circuits with those obtained by using the state-of-the-art circuit cutting technique *CutQC*. Further, to demonstrate the versatility and compatibility of *FragQC* across diverse quantum hardware platforms, we systematically considered multiple quantum hardware configurations and executed multiple benchmark circuits on each platform, aiming to validate the performance and applicability of *FragQC* in varied quantum computing environments.

Experimental setup. The overall experimental setup is mentioned below:

- The system configuration on which all the experiments have been performed in Python 3.11.1 is AMD EPYC 7B13 (x86_64) octacore on KVM, CPU 2.5 GHz, RAM 62.8Gi Usable, and x86_64 Ubuntu 22.04.2 LTS Operating System.
- For the GA-based approach, we initialize the partition vector randomly, comprising an equal distribution of '0's and '1's. Each experiment is repeated 30 times, and the optimal outcome is selected from the set of experiments for analysis.
- The quantum annealing-based approach has been implemented by using the hybrid binary quadratic model (BQM) of D-Wave, with an annealing time of 20 μs to solve the circuit partitioning QUBO.
- We have used noisy simulation to conduct our experiments and have considered the noise profiles of ibm_sherbrooke, a 127-qubit device with an Eagle-r3 processor. It has a median two-qubit gate error rate of $7.468e^{-3}$ and a median single-qubit gate error rate of $2.213e^{-4}$. The relaxation and coherence times associated with ibm_sherbrooke are 267.34 μs and 190.95 μs respectively.
- In order to demonstrate generalizability, we have run our benchmark circuits on the following IBM hardware: Athens, Belem, Hanoi, Washington, Singapore, Kolkata, and Mumbai.

4.1. Performance evaluation of all the partitioning approaches in FragQC

Results: Fig. 6(a) provides the cost of the cut selected by the specified algorithms, and Fig. 6(b) the cut-net or cut-size. The red bar represents *h-METIS*, the green one is for quantum annealing using the D-Wave systems, the orange bar represents *Mt-KaHyPar* and the blue one is for the proposed classical GA-inspired approach. It is evident from these results that the cut selected by *h-METIS* has a much higher cost and cut size, than the other two approaches. However, We cannot identify a clear winner among the classical GA-inspired approach, *Mt-KaHyPar* and the quantum annealing-based approach from the benchmark circuit execution. Hence, going forward, we should use the GA-inspired approach, *Mt-KaHyPar*, and QA-based approaches for our circuits and *FragQC* reports the cut that has the lowest cost associated with it.

In order to compare the quality of solutions provided by our proposed GA-inspired approach and *Mt-KaHyPar*, we have considered quantum circuits of varying sizes. In Fig. 7, we have plotted the cost of solutions of quantum circuits of various depths and widths with a purple bar for *Mt-KaHyPar* and red for the GA-inspired approach. Here, the X-axis represents an ordered pair of depth and width. Multiple circuits of each pair are executed and box-plots of their costs are presented. From Fig. 7, we can infer that the proposed GA-inspired approach is generating a better solution for the majority of the circuits.

Present-day quantum circuits are small in size, leading to graphs with a maximum node count typically in the range of a few hundred. Thus, the time-to-solution for most of the graph partitioning approaches used here remains reasonably small. Despite the manageable computational times, the application of these techniques has the potential to significantly improve the overall fidelity of quantum circuits. Hence, we prioritize solution quality over time-to-solution. However, we have performed a comparative study on the time-to-solution for different partitioning algorithms which is discussed in Appendix C.

4.2. Fidelity obtained with FragQC versus CutQC

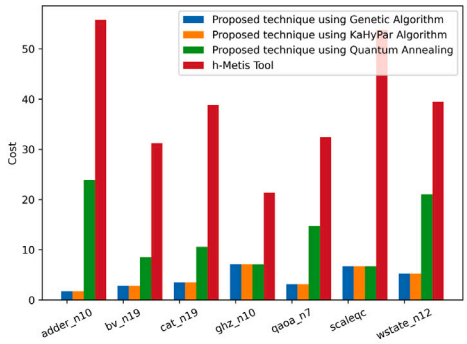
Let us consider the circuit shown in Fig. 5(a). Its corresponding doubly weighted graph is given in Fig. 5(b). In order to apply *FragQC* on this circuit, the error-balanced Min-cut finder reports the partition as shown in Fig. 8. In this case, the blue vertices form one sub-circuit, while the red ones the other. For this cut, only two edges, each having weight of 1, are cut, thus the cut size is 2 and the overall cost for the cut is 8.239. The two sub-circuits are shown in Fig. 9. These two sub-circuits can then be executed in the quantum hardware and the final outcome can be obtained through classical reconstruction.

Despite the fact that *FragQC* is a hardware-sensitive tool, it can easily be used with gate-based quantum hardware of any technology and size. We have used our tool *FragQC* to cut the quantum circuit whenever the success probability was below a certain user-defined threshold fidelity and executed the smaller sub-circuits on quantum hardware. We have calculated the fidelity compared to the ideal simulation. For two discrete probability distributions, $P = (p_1, \ldots, p_k)$ and $Q = (q_1, \ldots, q_k)$, the Hellinger distance is defined as:

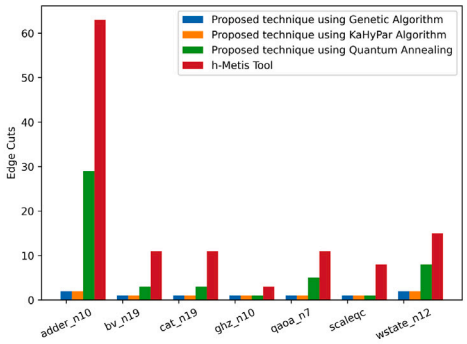
$$H(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{k}(\sqrt{p_i} - \sqrt{q_i})^2} \quad (13)$$

Hellinger fidelity is written as $(1 - H^2(P,Q))^2$, which is equivalent to standard classical fidelity.

As mentioned earlier, the determination of the threshold for *FragQC* for benchmark circuits is specified by the user. Albeit, we try to show through numerical analysis that it would not guarantee that the fidelity for benchmark circuits would always increase if the threshold level is set very high. We have taken four example benchmark circuits to analyze the fidelity of the circuit using *FragQC*, while we vary the success probability threshold. In Fig. 10, we portray the numerical results



(a) Cost of the cuts selected by *h-METIS* vs Mt-KaHyPar vs Proposed QA vs Proposed GA-inspired approach.



(b) Cut-net(cut size) size of the chosen cuts by *h-METIS* vs Mt-KaHyPar vs Proposed QA vs Proposed GA-inspired approach.

Fig. 6. Cost and cut-net (cut-size) for the selected cut by *h-METIS* (red), proposed QA (green), proposed GA-inspired approach (blue) and *Mt-KaHyPar* (orange) for different benchmark circuits. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

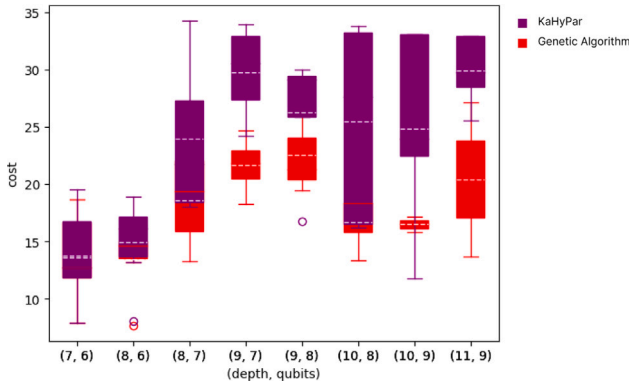


Fig. 7. Comparison of cost for *Mt-KaHyPar* vs. GA-inspired. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

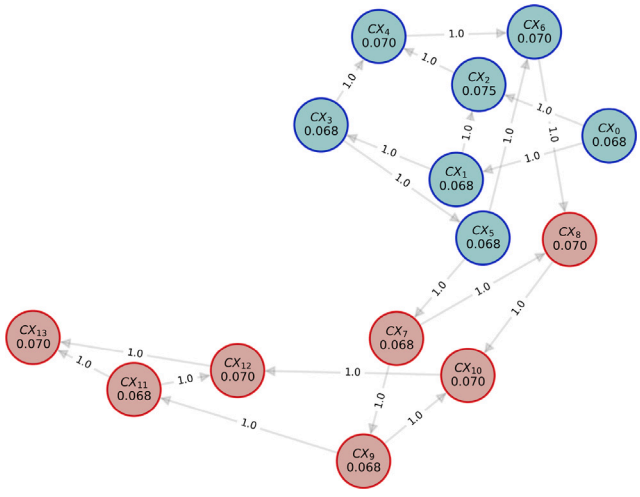


Fig. 8. Balanced bi-partitioning of the graph shown in Fig. 5(b) by using error balanced min-cut finding algorithm. The vertices of the subgraphs for the two partitions are marked in blue and red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

from which we can conclude that increasing the success probability threshold would not ensure an increase in fidelity.

Table 1
Fidelity for benchmark circuits on ibm_sherbrooke.

Benchmark circuit	Width	Depth	Fidelity		
			Without cut	*CutQC*	*FragQC*
Efficient SU	8	12	0.849	0.864	0.879
GHZ_n10	10	11	0.738	0.735	0.799
Adder n_10	10	108	0.735	0.734	0.738
BV_n19	20	22	0.497	0.534	0.598
Cat_n24	24	25	0.394	0.491	0.558
Deutsch Jozsa	20	12	0.792	0.808	0.814
Amplitude estimation	10	39	0.895	0.929	0.949
Quantum phase estimation - exact	6	15	0.850	0.895	0.919
Graphstate	8	5	0.980	0.987	0.989
QAOA	8	19	0.792	0.854	0.921
GHZ	12	12	0.678	–	0.726

For the purpose of a comparative study, we have leveraged *CutQC* (Tang et al., 2021) through the circuit knitting toolbox. We have also directly executed the circuits on *IBM_Sherbrooke* without cutting the circuits. The results obtained are shown in Table 1. For all the circuits, we observed better fidelity for *FragQC* over *CutQC* and direct execution without cutting. On average, we obtained 13.38% better fidelity compared to direct execution without cutting the circuit and 7.88% fidelity gain over *CutQC*.

Since quantum circuit fragmentation and knitting involve several measurement operations, we wanted to eradicate errors that occurred due to noisy measurement operations. For this purpose, we have used IBM's measurement error mitigation process to reduce the readout errors from the probability distribution output generated by each sub-circuit after hardware execution. Table 2 displays the fidelity of the benchmark circuits when executed on *FragQC* along with measurement error mitigation. We have also integrated measurement error mitigators with *CutQC* and direct execution and compared their corresponding fidelity. The fidelity obtained using *FragQC* is approximately 8.99% better than the *CutQC* with error mitigation.

We have also executed various benchmark circuits on multiple quantum hardware devices, as shown in Table 3. This comparison demonstrates that our technique is easily adaptable and generalizable across diverse quantum hardware configurations.

Lastly, for the machine learning-based methodology adopted by *i-QER* (Basu et al., 2022), it is necessary to generate training data and train a machine learning model for each distinct quantum hardware instance, which imposes limitations on the generalizability of the approach. In contrast, our proposed technique captures only the essential parameters, such as T1, T2, and gate execution time, from the

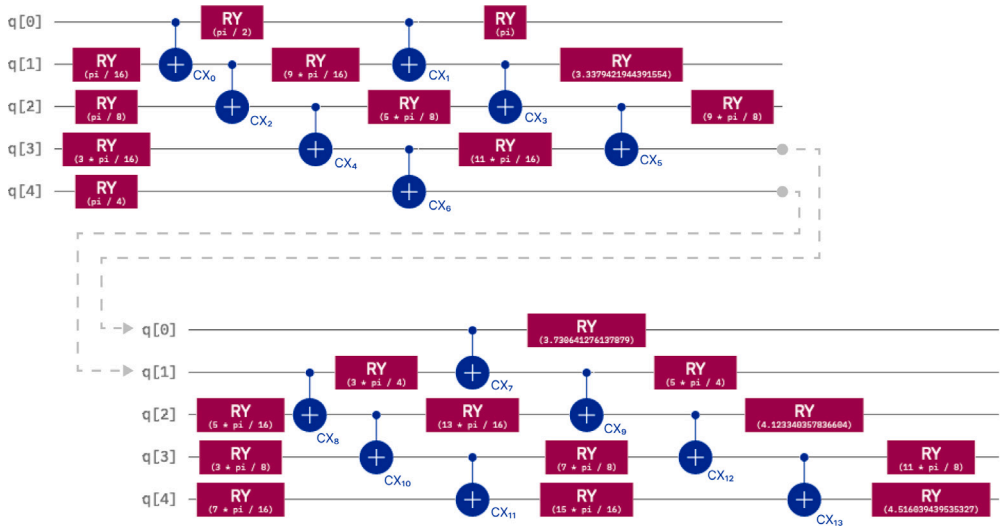


Fig. 9. Two sub-circuits corresponding to the two sub-graphs of Fig. 8 produced by our error-balanced min-cut finding algorithm with the threshold fidelity of 0.7.

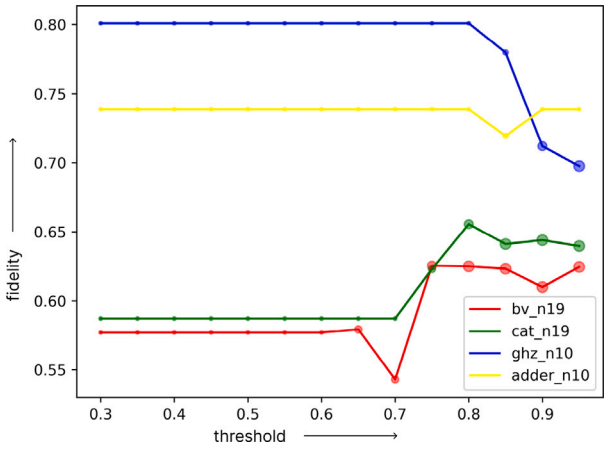


Fig. 10. Fidelity obtained by using *FragQC* on four benchmark circuits, with varying thresholds of success probability.

Table 2
Fidelity for benchmark circuits with measurement error mitigation.

Benchmark circuits	Width	Depth	Fidelity		
			Without cut	*CutQC*	*FragQC*
Efficient_SU	8	12	0.859	0.874	0.898
Ghz n10	10	11	0.813	0.951	0.986
Adder n10	10	108	0.783	0.813	0.957
BV n19	20	22	0.638	0.861	0.963
Cat n24	24	25	0.585	0.899	0.989

Table 3
Fidelity of the circuit for different benchmark quantum circuits across seven quantum backends of IBM by using *FragQC*.

Circuit name	Fidelity on quantum hardware						
	Athens	Belem	Hanoi	Washington	Singapore	Kolkata	Mumbai
Grover n3	0.997	0.996	0.996	0.997	0.994	0.998	0.995
QAOA n6	0.980	0.960	0.965	0.970	0.958	0.979	0.968
Wstate_n6	0.847	0.724	0.878	0.840	0.642	0.902	0.693
GHZ n10	0.847	0.767	0.846	0.6776	0.6116	0.843	0.707
BV n10	0.735	0.598	0.761	0.724	0.672	0.862	0.768

quantum hardware and calculates the error probability using Eq. (6) in this article. This approach facilitates efficient bi-partitioning of the circuit, thereby rendering our technique easily adaptable and generalizable across diverse quantum hardware configurations. *i-QER* makes an exhaustive search to find the most efficient cut, which adds an exponential cost. Additionally, it limits the allowed cut size to restrict the exponential growth of the possible search space. This limitation can restrict *i-QER* from cutting large-scale circuits. However, *FragQC* overcomes these limitations by an error-efficient cut searcher which minimizes the objective function given in Eq. (9) that balances errors in each partition while minimizing the cut size.

5. Discussion and conclusion

In this paper, our aim is to reduce errors in a quantum circuit through quantum circuit fragmentation. Although quantum circuit fragmentation presents a viable approach for error reduction, it comes with several associated limitations. These include exponentially high classical post-processing costs and the potential loss of entanglement during output reconstruction. Therefore, the approach of fragmenting quantum circuits should be employed with caution, i.e., only when it is necessary. The classical post-processing cost of fragmentation depends on the number of edges (qubits) cut. To reduce error in a quantum circuit, we have proposed a method for balanced bi-partitioning of a doubly weighted graph constructed from a quantum circuit, facilitating quantum circuit fragmentation. The technique considers both the hardware noise and the cut size. Multiple graph partitioning techniques, including h-metis, Mt-KaHyPar, and quantum annealing, among others, have been investigated for implementing this process.

During the experimental evaluation, classical genetic algorithm-inspired techniques, MT-KaHyPar, and quantum annealing-based approaches demonstrated superior performance in terms of solution quality. Our proposed tool, FragQC, incorporates these three techniques for partitioning a quantum circuit. We have also exhibited through our proposed tool, *FragQC*, that the fidelity of the benchmark circuits has significantly improved with a GA-inspired method or a quantum annealing method within it for solving the problem in comparison with the existing circuit fragmentation methods. Additionally, we have leveraged measurement error mitigation along with our proposed tool *FragQC* and observed significant improvements in fidelity over direct execution and *CutQC*. More specifically, we have achieved a significant improvement in fidelity through our approach, demonstrating an increase of 13.38% compared to direct execution without cutting

Table C.4
Time-to-solution (secs.) of the four graph partitioning methods in *FragQC*.

# nodes	Quantum annealing	GA-inspired	*Mt-KaHyPar*	*hMetis*
7	5.005423	0.035283	0.020307	0.014518
8	5.006752	0.057306	0.024547	0.014068
9	5.005797	0.059235	0.016537	0.014929
11	5.009468	0.099897	0.031525	0.089972
12	5.003700	0.156984	0.057096	0.104111
13	5.009785	0.110163	0.019492	0.094653
14	5.007853	0.239869	0.037405	0.122351
15	5.004134	0.129511	8.309534	0.111695
16	5.008116	0.349666	8.867076	0.163415
17	5.008478	0.226014	0.080664	0.152313
18	5.005322	0.788631	0.103441	0.183569
19	5.009067	0.817232	0.095283	0.187192
20	5.001709	2.357681	0.154360	0.167579
21	5.009955	0.948268	0.101810	0.176621
22	5.004664	1.398256	0.086735	0.215363
23	5.007342	3.686684	0.116055	0.212205
24	5.004970	1.433570	0.121892	0.292694
25	5.009295	5.978026	0.140706	0.225442
26	5.009387	3.732531	0.125497	0.239036
27	5.002463	5.989513	0.112075	0.284995
28	5.001034	5.981906	0.129976	0.286710
29	5.008552	6.896825	0.156014	0.305733
30	5.005878	2.836488	0.143351	0.278830
31	5.004331	6.745825	0.110312	0.283720
32	5.009959	6.580789	17.081148	0.349797
33	5.006193	8.008221	0.159010	0.311937
34	5.005946	7.821375	17.439122	0.303703
36	5.002543	7.213571	0.162318	0.317969
37	5.004499	9.579230	0.180068	0.477049
39	5.008275	10.224532	0.185538	0.369885

the circuit. Additionally, our method surpasses the state-of-the-art ILP-based approach by 7.88% for the benchmark circuits. To ensure the generalizability of the tool, we have executed multiple benchmark circuits on eight different quantum hardware.

Therefore, *FragQC* provides a hybrid quantum computing strategy. It is a robust method and can be implemented on any gate-based hardware. Since there may be multiple numbers of sub-circuits, these may be implemented and run on different hardware in parallel (Bhoumik et al., 2023; Ferrari et al., 2021; Chatterjee et al., 2022; Caleffi et al., 2022; Cacciapuoti et al., 2020; Illiano et al., 2022) to minimize the run-time complexity of our tool and pave the pathway towards hybrid distributed quantum computation. This may be explored in the future, along with the trade-off in the time for reconstruction of the results of the sub-circuits.

CRediT authorship contribution statement

Saikat Basu: Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Arnav Das:** Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation. **Amit Saha:** Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Amlan Chakrabarti:** Writing – review & editing, Validation, Supervision, Project administration, Formal analysis. **Susmita Sur-Kolay:** Writing – review & editing, Visualization, Validation, Supervision, Resources, Project administration, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I Have shared the code at the attached file.

Appendix A. Algorithm for cost function calculation

Algorithm 2 helps us in estimating the cost function of Eq. (9). The procedure, as outlined in Algorithm 2, includes determining the weights of vertices within each partition and employing Algorithm 3 (referred to as the Cut Size Calculator), to compute the size of the partition. Once this cost of the initial partition vector is computed, it is stored as $MinCost$.

Algorithm 2 Calculating the cost of a bi-partition

Input: $cutSize$, $partitionVector$, $vertexWeight$, N
▷ N = number of vertices
Output: Cost
function $CostCalculator(partitionVector)$
$WeightP2 \Leftarrow 0$
$totalWeight \Leftarrow 0$
for $i \leftarrow 1$ to N **do**
$totalWeight = totalWeight + vertexWeight[i]$
$WeightP2 = WeightP2 + vertexWeight[i] \times partitionVector[i]$
end for
$WeightP1 = totalWeight - WeightP2$
$Cost = CutSize(partitionVector) \times (\frac{1}{WeightP1} + \frac{1}{WeightP2})$
return Cost
end function

Algorithm 3 Calculating size of min-cut

Input: $partitionVector$, $edges$, $edgeWeight$
Output: $cutSize$
/* $partitionVector$ is '0' or '1' for vertices $(v_1, v_2, ..v_N)$ in Partition1 or Partition2 respectively, edge $e_{k,l}$ have endpoints v_k and v_l, $edgeWeight$ is the weight of each edge */

function $CutSize(partitionVector)$
$cutSize \Leftarrow 0$
while $edges \neq NULL$ **do** ▷ We have considered all the edges
$cutSize = cutSize + edgeWeight[e_{i,j}] \times (partitionVector[i] - partitionVector[j])^2$ ▷ Edge $e_{i,j}$ is an edge between vertex v_i and v_j
end while
return $cutSize$
end function

Appendix B. Algorithm for crossover operation

Algorithm 4 implements the single-point crossover operation. It takes as input two partition vectors and a random number c_1 ($0 \leq c_1 \leq 1$) to identify the crossover point.

Appendix C. Comparative study of time-to-solution

A comparative study of time-to-solution of different graph partitioning techniques was carried out by generating multiple random quantum circuits with nodes (two-qubit gates) ranging from 7 to 39 and executing them. The time-to-solution for different graph partitioning algorithms are shown in Table C.4. The analysis of results obtained reveals that *Mt-KaHyPar* consistently exhibits the lowest time-to-solution across the majority of cases. In contrast, the time-to-solution for quantum annealing remains independent of the number of nodes, consistently hovering around 5 s for all conducted experiments. Notably, our proposed GA-inspired approach, while demonstrating better

Algorithm 4 Crossover algorithm

Input: $Vector1$, $Vector2$, N
Output: $FinalVector$
  function $crossover(Vector1, Vector2, N, c_1)$
    $FinalVector \Leftarrow 0$
    for $i \leftarrow 0$ to $c_1 N$ **do**
      $FinalVector[i] = Vector1[i]$
    end for
    for $j \leftarrow 0$ to $N(1 - c_1)$ **do**
      $FinalVector[cj + 1] = Vector2[j]$
    end for
    return $FinalVector$
  end function

performance in terms of quality of solution, exhibits a relatively higher time-to-solution. These findings contribute to a nuanced understanding of the computational efficiency of the respective techniques under consideration.

References

Ayral, T., Le Régent, F.-M., Saleem, Z., Alexeev, Y., Suchara, M., 2020. Quantum divide and compute: Hardware demonstrations and noisy simulations. In: 2020 IEEE Computer Society Annual Symposium on VLSI. ISVLSI, pp. 138–140. http://dx.doi.org/10.1109/ISVLSI49217.2020.00034.

Ayral, T., Régent, F.-M.L., Saleem, Z., Alexeev, Y., Suchara, M., 2021. Quantum divide and compute: Exploring the effect of different noise sources. arXiv:2102.03788.

Baek, C., Ostuka, T., Tarucha, S., Choi, B.-S., 2019. Density matrix simulation of quantum error correction codes for near-term quantum devices. Quantum Sci. Technol. 5 (1), 015002. http://dx.doi.org/10.1088/2058-9565/ab5887.

Basu, S., Saha, A., Chakrabarti, A., Sur-Kolay, S., 2022. I-QER: An intelligent approach towards quantum error reduction. ACM Trans. Quantum Comput. 3 (4), http://dx.doi.org/10.1145/3539613.

Bhoumik, D., Majumdar, R., Saha, A., Sur-Kolay, S., 2023. Distributed scheduling of quantum circuits with noise and time optimization. arXiv:2309.06005.

Born, M., Fock, V., 1928. Beweis des adiabatensatzes. Z. fur Phys. 51 (3), 165–180. http://dx.doi.org/10.1007/BF01343193.

Bravyi, S., Smith, G., Smolin, J.A., 2016. Trading classical and quantum computational resources. Phys. Rev. X 6 (2), http://dx.doi.org/10.1103/physrevx.6.021043, URL http://dx.doi.org/10.1103/PhysRevX.6.021043.

Bruzewicz, C.D., Chiaverini, J., McConnell, R., Sage, J.M., 2019. Trapped-ion quantum computing: Progress and challenges. Appl. Phys. Rev. 6 (2), 021314. http://dx.doi.org/10.1063/1.5088164.

Bui, T.N., Moon, B.R., 1996. Genetic algorithm and graph partitioning. IEEE Trans. Comput. 45 (7), 841–855. http://dx.doi.org/10.1109/12.508322.

Cacciapuoti, A.S., Caleffi, M., Tafuri, F., Cataliotti, F.S., Gherardini, S., Bianchi, G., 2020. Quantum internet: Networking challenges in distributed quantum computing. IEEE Netw. 34 (1), 137–143. http://dx.doi.org/10.1109/MNET.001.1900092.

Cai, Z., Babbush, R., Benjamin, S.C., Endo, S., Huggins, W.J., Li, Y., McClean, J.R., O'Brien, T.E., 2023. Quantum error mitigation. Rev. Mod. Phys. 95, 045005. http://dx.doi.org/10.1103/RevModPhys.95.045005, URL https://link.aps.org/doi/10.1103/RevModPhys.95.045005.

Caleffi, M., Amoretti, M., Ferrari, D., Cuomo, D., Illiano, J., Manzalini, A., Cacciapuoti, A.S., 2022. Distributed quantum computing: a survey. arXiv:2212.10609.

Campbell, E.T., Terhal, B.M., Vuillot, C., 2017. Roads towards fault-tolerant universal quantum computation. Nature 549 (7671), 172–179. http://dx.doi.org/10.1038/nature23460.

Chatterjee, T., Das, A., Mohtashim, S.I., Saha, A., Chakrabarti, A., 2022. Qurzon: A prototype for a divide and conquer-based quantum compiler for distributed quantum systems. SN Comput. Sci. 3 (4), http://dx.doi.org/10.1007/s42979-022-01207-9.

Deb, K., Sindhya, K., Hakanen, J., 2016. Multi-objective optimization. In: Decision Sciences. CRC Press, pp. 161–200.

Farhi, E., Goldstone, J., Gutmann, S., Sipser, M., 2000. Quantum computation by adiabatic evolution. arXiv:quant-ph/0001106.

Ferrari, D., Cacciapuoti, A.S., Amoretti, M., Caleffi, M., 2021. Compiler design for distributed quantum computing. IEEE Trans. Quantum Eng. 2, 1–20. http://dx.doi.org/10.1109/TQE.2021.3053921.

Fiduccia, C., Mattheyses, R., 1982. A linear-time heuristic for improving network partitions. In: 19th Design Automation Conference. pp. 175–181. http://dx.doi.org/10.1109/DAC.1982.1585498.

Fowler, A.G., Mariantoni, M., Martinis, J.M., Cleland, A.N., 2012. Surface codes: Towards practical large-scale quantum computation. Phys. Rev. A 86, 032324. http://dx.doi.org/10.1103/PhysRevA.86.032324, URL https://link.aps.org/doi/10.1103/PhysRevA.86.032324.

Gokhale, P., Baker, J.M., Duckering, C., Brown, N.C., Brown, K.R., Chong, F.T., 2019. Asymptotic improvements to quantum circuits via qutrits. In: Proceedings of the 46th International Symposium on Computer Architecture. ISCA '19, Association for Computing Machinery, New York, NY, USA, pp. 554–566. http://dx.doi.org/10.1145/3307650.3322253.

Gottesman, D., 1999. Fault-Tolerant quantum computation with higher-dimensional systems. In: Williams, C.P. (Ed.), Quantum Computing and Quantum Communications. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 302–313.

Graham, T., Song, Y., Scott, J., Poole, C., Phuttitarn, L., Jooya, K., Eichler, P., Jiang, X., Marra, A., Grinkemeyer, B., Kwon, M., Ebert, M., Cherek, J., Lichtman, M., Gillette, M., Gilbert, J., Bowman, D., Ballance, T., Campbell, C., Saffman, M., 2022. Multi-qubit entanglement and algorithms on a neutral-atom quantum computer. Nature 604, 457–462. http://dx.doi.org/10.1038/s41586-022-04603-6.

Grassl, M., Geiselmann, W., Beth, T., 1999. Quantum Reed—Solomon codes. In: Fossorier, M., Imai, H., Lin, S., Poli, A. (Eds.), Applied Algebra, Algebraic Algorithms and Error-Correcting Codes. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 231–244.

Illiano, J., Caleffi, M., Manzalini, A., Cacciapuoti, A.S., 2022. Quantum internet protocol stack: A comprehensive survey. Comput. Netw. 213, 109092. http://dx.doi.org/10.1016/j.comnet.2022.109092, URL https://www.sciencedirect.com/science/article/pii/S1389128622002250.

Karypis, G., Aggarwal, R., Kumar, V., Shekhar, S., 1997. Multilevel hypergraph partitioning: Application in VLSI domain. In: Proceedings of the 34th Annual Design Automation Conference. DAC '97, Association for Computing Machinery, New York, NY, USA, pp. 526–529. http://dx.doi.org/10.1145/266021.266273.

Karypis, G., Kumar, V., 1998a. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. 20 (1), 359–392. http://dx.doi.org/10.1137/S1064827595287997, arXiv:https://doi.org/10.1137/S1064827595287997.

Karypis, G., Kumar, V., 1998b. Multilevelk-way partitioning scheme for irregular graphs. J. Parallel Distrib. Comput. 48 (1), 96–129. http://dx.doi.org/10.1006/jpdc.1997.1404, URL https://www.sciencedirect.com/science/article/pii/S0743731597914040.

Karypis, G., Kumar, V., 1999. Multilevel K-way hypergraph partitioning. In: Proceedings of the 36th Annual ACM/IEEE Design Automation Conference. DAC '99, Association for Computing Machinery, New York, NY, USA, pp. 343–348. http://dx.doi.org/10.1145/309847.309954.

Kernighan, B.W., Lin, S., 1970. An efficient heuristic procedure for partitioning graphs. Bell Syst. Tech. J. 49 (2), 291–307. http://dx.doi.org/10.1002/j.1538-7305.1970.tb01770.x.

Kim, C., Park, K.D., Rhee, J.-K., 2020. Quantum error mitigation with artificial neural network. IEEE Access 8, 188853–188860. http://dx.doi.org/10.1109/ACCESS.2020.3031607.

Koch, J., Yu, T.M., Gambetta, J., Houck, A.A., Schuster, D.I., Majer, J., Blais, A., Devoret, M.H., Girvin, S.M., Schoelkopf, R.J., 2007. Charge-insensitive qubit design derived from the cooper pair box. Phys. Rev. A 76, 042319. http://dx.doi.org/10.1103/PhysRevA.76.042319, URL https://link.aps.org/doi/10.1103/PhysRevA.76.042319.

Laflamme, R., Miquel, C., Paz, J.P., Zurek, W.H., 1996. Perfect quantum error correcting code. Phys. Rev. Lett. 77, 198–201. http://dx.doi.org/10.1103/PhysRevLett.77.198, URL https://link.aps.org/doi/10.1103/PhysRevLett.77.198.

Leuenberger, M.N., Loss, D., 2001. Quantum computing in molecular magnets. Nature 410 (6830), 789–793. http://dx.doi.org/10.1038/35071024.

Li, A., Stein, S., Krishnamoorthy, S., Ang, J., 2022. QASMBench: A low-level QASM benchmark suite for NISQ evaluation and simulation. arXiv:2005.13018.

Loss, D., DiVincenzo, D.P., 1998. Quantum computation with quantum dots. Phys. Rev. A 57, 120–126. http://dx.doi.org/10.1103/PhysRevA.57.120, URL https://link.aps.org/doi/10.1103/PhysRevA.57.120.

Lucas, A., 2014. Ising formulations of many NP problems. Front. Phys. 2, http://dx.doi.org/10.3389/fphy.2014.00005.

Majumdar, R., Basu, S., Mukhopadhyay, P., Sur-Kolay, S., 2016. Error tracing in linear and concatenated quantum circuits. arXiv:1612.08044.

Majumdar, R., Madan, D., Bhoumik, D., Vinayagamurthy, D., Raghunathan, S., Sur-Kolay, S., 2021. Optimizing ansatz design in QAOA for Max-cut. arXiv:2106.02812.

Nautrup, H.P., Delfosse, N., Dunjko, V., Briegel, H.J., Friis, N., 2019. Optimizing quantum error correction codes with reinforcement learning. Quantum 3, 215. http://dx.doi.org/10.22331/q-2019-12-16-215.

Nielsen, M.A., Chuang, I.L., 2010. Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, http://dx.doi.org/10.1017/CBO9780511976667.

Nielsen, M.A., Chuang, I.L., 2011. Quantum Computation and Quantum Information: 10th Anniversary Edition, tenth edition Cambridge University Press, USA.

Peng, T., Harrow, A.W., Ozols, M., Wu, X., 2020. Simulating large quantum circuits on a small quantum computer. Phys. Rev. Lett. 125, 150504. http://dx.doi.org/10.1103/PhysRevLett.125.150504, URL https://link.aps.org/doi/10.1103/PhysRevLett.125.150504.

Perlin, M.A., Saleem, Z.H., Suchara, M., Osborn, J.C., 2021a. Quantum circuit cutting with maximum likelihood tomography. arXiv:2005.12702.

Perlin, M.A., Saleem, Z.H., Suchara, M., Osborn, J.C., 2021b. Quantum circuit cutting with maximum-likelihood tomography. npj Quantum Inf. 7 (1), 64. http://dx.doi.org/10.1038/s41534-021-00390-6.

Preskill, J., 2018. Quantum computing in the NISQ era and beyond. Quantum 2, 79. http://dx.doi.org/10.22331/q-2018-08-06-79.

Quetschlich, N., Burgholzer, L., Wille, R., 2023. MQT Bench: Benchmarking software and design automation tools for quantum computing. Quantum 7, 1062. http://dx.doi.org/10.22331/q-2023-07-20-1062.

Saha, A., Majumdar, R., Saha, D., Chakrabarti, A., Sur-Kolay, S., 2022. Asymptotically improved circuit for a *d*-ary grover's algorithm with advanced decomposition of the *n*-qudit toffoli gate. Phys. Rev. A 105, 062453. http://dx.doi.org/10.1103/PhysRevA.105.062453, URL https://link.aps.org/doi/10.1103/PhysRevA.105.062453.

Shor, P.W., 1995. Scheme for reducing decoherence in quantum computer memory. Phys. Rev. A 52, R2493–R2496. http://dx.doi.org/10.1103/PhysRevA.52.R2493, URL https://link.aps.org/doi/10.1103/PhysRevA.52.R2493.

Steane, A.M., 1996. Error correcting codes in quantum theory. Phys. Rev. Lett. 77, 793–797. http://dx.doi.org/10.1103/PhysRevLett.77.793, URL https://link.aps.org/doi/10.1103/PhysRevLett.77.793.

Tang, W., Martonosi, M., 2022. Scaleqc: A scalable framework for hybrid computation on quantum and classical processors. arXiv:2207.00933.

Tang, W., Tomesh, T., Suchara, M., Larson, J., Martonosi, M., 2021. Cutqc: Using small quantum computers for large quantum circuit evaluations. In: Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. In: ASPLOS 2021, Association for Computing Machinery, New York, NY, USA, pp. 473–486. http://dx.doi.org/10.1145/3445814.3446758.

Terhal, B.M., 2015. Quantum error correction for quantum memories. Rev. Mod. Phys. 87, 307–346. http://dx.doi.org/10.1103/RevModPhys.87.307, URL https://link.aps.org/doi/10.1103/RevModPhys.87.307.

Ushijima-Mwesigwa, H., Negre, C.F.A., Mniszewski, S.M., 2017. Graph partitioning using quantum annealing on the D-Wave system. arXiv:1705.03082.

Wootton, J.R., 2020. Benchmarking near-term devices with quantum error correction. Quantum Sci. Technol. 5 (4), 044004. http://dx.doi.org/10.1088/2058-9565/aba038.