IEEE Transactions on Quantum Engineering

Received 23 December 2022; revised 9 April 2023; accepted 26 April 2023; date of publication 1 May 2023; date of current version 16 June 2023.

Digital Object Identifier 10.1109/TQE.2023.3272023

Fuzzy-Based Balanced Partitioning Under Capacity and Size-Tolerance Constraints in Distributed Quantum Circuits

JIN-TAI YAN (Member, IEEE)
Graduate Institute of Sound Technology, Tainan National University of the Arts, Tainan 720, Taiwan

ABSTRACT It is important for the design of a distributed quantum circuit (DQC) to minimize the communication cost in k-way balanced partitioning. In this article, given an original quantum circuit (QC), a partitioning number k, the maximum capacity δ inside each partition, and the maximum size tolerance γ between two partitions, a new k-way (δ, γ)-balanced partitioning problem can be formulated as a k-way partitioning problem under the capacity constraint δ and the size-tolerance constraint γ, and a fuzzy-based partitioning algorithm can be proposed to minimize the communication cost in k-way (δ, γ)-balanced partitioning for a DQC design. First, an edge-weighted connection graph can be constructed from the gates in a given QC. Furthermore, based on the estimation of the probabilistic connection strength between two vertices in the connection graph and the *initial k-way* partitioning result in the connection graph, the fuzzy memberships on k clusters can be generated in fuzzy k-means graph clustering. Finally, based on the fuzzy memberships on k clusters in the connection graph, the maximum capacity inside each partition, and the maximum size tolerance between two partitions, all the vertices in the connection graph can be assigned onto k partitions to minimize the communication cost in k-way (δ, γ)-balanced partitioning. Compared with Daei's recursive Kernighan–Lin-based algorithm in four-way balanced partitioning, the experimental results show that the proposed fuzzy-based partitioning algorithm with three size-tolerance constraints $\gamma = 1$, $\gamma = 2$, and $\gamma = 3$ can use 58.3%, 61.3%, and 64.5% of CPU time to reduce 16.1%, 21.2%, and 24.6% of the communication cost for the eight tested circuits on the average, respectively. Compared with the modified partitioning algorithm from Dadkhah's partitioning algorithm in three-way, four-way, or five-way balanced partitioning, the experimental results show that the proposed fuzzy-based partitioning algorithm with the size-tolerance constraint $\gamma = 3$ can use 35.0% of CPU time to reduce 11.1% of the communication cost for the eight tested circuits on the average, respectively.

INDEX TERMS Balanced partitioning, communication cost, distributed quantum circuit (DQC), fuzzy k-means graph clustering (FKGC).

I. INTRODUCTION

Quantum computing [1] can use the principles of quantum mechanics to efficiently solve some specific problems by designing quantum algorithms. It is known that the quantum algorithms can be described by using quantum circuits (QCs) [2], [3] on an ideal quantum computer. To execute some large-scale quantum algorithms on a quantum computer, many physical qubits must be required in the QCs. Due to the technology constraints, the number of the available quantum bits (qubits) in a fabricated quantum device [4] is limited. Hence, the existence of these technology constraints will lead to the emergence of distributed quantum computing. In the design of a distributed quantum circuit (DQC), a set of smaller scale quantum devices must be constructed for large-scale quantum algorithms, and there can be some communication methods among smaller scale quantum devices.

In the concept of using DQC designs, Cleve and Buhrman [5] first studied the quantum communication among remote quantum devices. As the input data are distributed among remote quantum devices, quantum entanglement can be used for communication. Furthermore, Cirac et al. [6] showed that the use of maximally entangled states can be advantageous for a large number of quantum devices using ideal quantum channels. In addition, Beals et al. [7] also showed that each QC can be converted into a DQC design in a distributed quantum system.

For communication in DQC designs, Yepez [8] first presented a distributed architecture using two communication

methods. Basically, each qubit can be entangled to any number of qubits in quantum communication, and all the remote quantum devices can be connected by a set of classical channels in classical communication. In addition, Lo [9] examined the cost of the classical communication on DQC designs. Furthermore, Caleffi et al. [10], [11] stated that the communication among remote quantum devices can become faster by using the quantum Internet. In the design of the quantum Internet, teleportation can be used as the main strategy for information transmission.

For two-way partitioning in DQC designs, Yimsiriwattana and Lomonaco [12] first proposed the distributed model of Shor's algorithm [13]. In the distributed model, the global gates can be used to implement the DQC design, and teleportation can be used as the communication method. However, the partitioning of the used qubits cannot be determined to minimize the number of teleportations in the distributed model. Furthermore, based on the teleportation for communication, Van Meter et al. [14] proposed the distributed design of a two-qubit Vedral–Barenco–Ekert (VBE) carry-ripple adder onto two equal quantum devices. However, the teleportation cost in the distributed design can still be higher. Next, Zomorodi-Moghadam et al. [15] proposed a heuristic algorithm to reduce the communication cost between two partitions inside a DQC design. By finding the execution order of two-qubit controlled-NOT (CNOT) gates, the total number of teleportations between the two partitions can be reduced to minimize the teleportation cost between two distributed devices. In addition, Houshmand et al. [16] proposed an evolutionary algorithm, and Dadkhah et al. [17] proposed a genetic algorithm to minimize the communication cost between two partitions inside a DQC design. Recently, based on the connectivity matrix model of QCs, Ghodsollahee et al. [18] further proposed a two-phase algorithm to minimize the communication cost between two partitions inside a DQC design. However, these proposed algorithms do not consider the multiple-way partitioning on the minimization of the communication cost in a DQC design.

In multiple-way partitioning for a DQC design, Andrés-Martínez and Heunen [19] presented an automated method to distribute a QC over multiple devices. However, the minimization of the communication cost does not need to be considered for multiple partitions in a DQC design, and the assignment of the global gates in different partitions does not need to be further discussed in the proposed algorithm. Based on the construction of the bipartite graph for a given QC, Davarzani et al. [20] proposed a dynamic programming (DP) algorithm to partition the bipartite graph into some low-capacity QCs. However, any QC cannot be guaranteed to be presented as a bipartite graph, and the DP algorithm takes more execution time in multiple-way partitioning. Additionally, Daei et al. [21] also proposed an iterative Kernighan–Lin-based (KL-based) algorithm in a DQC design from a monolithic QC. In the proposed algorithm, the communication cost between multiple partitions inside a DQC design can be minimized. However, based on the utilization of the KL algorithm, the proposed recursive KL-based algorithm can only be used on the constrained number of partitions in a DQC design. Recently, based on the reordering result of a QC for the improvement of the execution time and the construction of a graph mode for a QC, Dadkhah et al. [22] proposed the genetic algorithm and the modified tabu-search algorithm to partition the graph model to obtain a DQC. However, the genetic algorithm and the modified tabu-search algorithm take more execution time in multiple-way partitioning.

The contributions of this article can be summarized as follows.

1) In a DQC design, a new k-way balanced partitioning (KBP) problem with two adjustable parameters on maximum capacity and maximum size tolerance can be formatted. Given an original QC, a partitioning number k, the maximum capacity δ inside each partition, and the maximum size tolerance γ among two partitions, an edge-weighted connection graph can be constructed from the gates in the given QC for k-way (δ, γ)-balanced partitioning under the capacity constraint δ and the size-tolerance constraint γ.
2) Based on the edge connections in an edge-weighted connection graph and the given partitioning number k, the probabilistic connection strength between two vertices in the connection graph can be estimated. Furthermore, the initial k-way partitioning result in the connection graph can be obtained by using a bottom-up clustering algorithm. Finally, based on the definition of the clustering distance between two vertices in the connection graph, the fuzzy memberships on k clusters in the connection graph can be generated in fuzzy k-means graph clustering (FKGC).
3) Based on the fuzzy memberships on k clusters in the connection graph, the capacity constraint δ inside each partition, and the size-tolerance constraint γ between two partitions, all the vertices in the connection graph can be assigned onto k partitions to minimize the communication cost in k-way (δ, γ)-balanced partitioning for a DQC design.

The rest of this article is organized as follows. Section II contains the motivation of the KBP in a DQC design and the formulation of the k-way (δ, γ)-balanced partitioning problem under the capacity constraint δ and the size-tolerance constraint γ in a DQC design. In Section III, based on the construction of an edge-weighted connection graph, the computation of the probabilistic connection strength between two vertices in the connection graph, the design of the FKGC, and the assignment of all the vertices in the connection graph onto k partitions, a fuzzy-based partitioning algorithm can be proposed to partition an original QC into k quantum subcircuits while minimizing the communication cost under the capacity constraint δ and the size-tolerance constraint γ in a DQC design. In Section IV, the experimental results in the proposed fuzzy-based partitioning algorithm can be listed and

compared with some published algorithms in k-way (δ, γ)-balanced partitioning for a DQC design. Finally, Section V concludes this article.

II. MOTIVATION AND PROBLEM FORMULATION

In quantum computing, it is necessary for the solution of a larger problem to use more quantum bits (qubits) inside a QC. Based on the superposition principle in QCs, the state $|\psi\rangle$ of a qubit can be represented by a unit vector in a Hilbert space labeled as $\alpha|0\rangle + \beta|1\rangle$, where $|0\rangle$ and $|1\rangle$ are the basis of space and α and β are two complex coefficients establishing $|\alpha|^2 + |\beta|^2 = 1$.

In general, a QC can consist of some quantum gates connected by a set of quantum wires for moving quantum data. Basically, a t-qubit quantum gate U can be defined and represented as a $2^t \times 2^t$ matrix. By performing a t-qubit quantum gate U on t quantum states, $|\psi_1\rangle$, $|\psi_2\rangle$, …, and $|\psi_t\rangle$, the outcome of the quantum gate U can be represented as the state $U(|\psi_1\rangle, |\psi_2\rangle, \ldots, |\psi_t\rangle)$. In addition, the controlled gate, controlled-U, with s-controlled qubits and t-qubit quantum gate U, operating on $(s+t)$-qubits, can be treated as a $(s+t)$-qubit gate. For example, the 1-controlled gate, CNOT, with one controlled qubit, and 1-qubit quantum gate, NOT, can be treated as a two-qubit gate.

Due to the limited capacity inside a QC in quantum computing technologies, a DQC design consisting of many small QCs on remote locations can be connected together to cooperatively solve a larger problem using the available capacities inside all the smaller QCs. In a DQC design, a local gate can be defined as a quantum gate where all qubits are located inside the same QC. On the other hand, a global gate can be defined as a quantum gate where all qubits are located inside some remote QCs. To implement the functionality of an original QC, the remote QCs in a DQC design must communicate with each other by sending the necessary qubits' information to each other using a quantum channel via teleportation.

Due to the limited capacity inside a small QC, the maximum capacity inside each small QC must be treated as the *capacity constraint* of one partitioned QC in a DQC design. Basically, the state of the controlled qubits inside some remote QCs must be sent to the operating qubit inside the other remote QC by communicating the qubits using teleportation. Since teleportation in a DQC design is a costly operation, the communication cost must be minimized in the partitioning process of a DQC design, that is, the number of the global gates must be minimized in the partitioning process of a DQC design. Additionally, to make use of the computing ability of the small QCs for a larger problem in a DQC design, the balance degree of the available partitions must be considered in a DQC design. Hence, the maximum size difference between two partitioned QCs in a DQC design must be treated as the *size-tolerance constraint* of all the partitions in a DQC design. Clearly, the smaller the maximum size difference between two partitioned QCs is, the higher the balance degree of a DQC design is. However, the communication cost in a DQC design will become more serious due to the higher balance degree of a DQC design. Under the acceptable balance degree of a DQC design, the allowable tolerant difference between two partitioned QCs in a DQC design can be used to reduce the communication cost in a DQC design.

A. MOTIVATION

In general, a given circuit can easily be bipartitioned into two balanced subcircuits in the KL algorithm. Hence, an original QC in a DQC design can be recursively partitioned by using the KL-based algorithm. However, the recursive KL-based algorithm can only partition an original QC into smaller QCs inside some constrained partitions, that is, the number of the partitions can be only constrained as 2^p in the recursive KL-based partitioning of a DQC design, where p is the number of recursions. Hence, it is necessary for a DQC design to consider the arbitrary partitions in the multiple-way partitioning of a DQC design.

On the other hand, it is known that the KL-based algorithm is a two-way balanced partitioning algorithm. Hence, the recursive KL-based algorithm is a multiple-way strictly balanced partitioning algorithm, that is, the size difference between two partitions is not larger than 1 in the recursive KL-based partitioning. Due to the strict size tolerance in the recursive KL-based partitioning, the strict balance will lead to the larger communication cost in a DQC design. If the acceptable size tolerance is considered in the multiple-way balanced partitioning, the communication cost can be further reduced in a DQC design.

B. PROBLEM FORMULATION

Initially, it is assumed that the communication cost between one controlled qubit and one operating qubit inside one gate can be set as 1. Given an original QC with n qubits $q_1, q_2, \ldots, q_n$ and m gates $U_1, U_2, \ldots, U_m$, a partitioning number k, the maximum capacity δ inside a partitioned QC, and the maximum size tolerance γ among two partitions, the k-way (δ, γ)-balanced partitioning can be formulated to partition the original QC into k partitions to minimize the communication cost among the k partitions with satisfying the capacity constraint δ inside each partition and the size-tolerance constraint γ between two partitions in a DQC design.

For the specification of an original QC with eight qubits q_1, $q_2, \ldots, q_8$ and 23 gates $U_1, U_2, \ldots, U_{23}$, in Fig. 1(a), it is clear that the two gates U_{19} and U_{22} are 2-controlled gates and the other gates are 1-controlled gates. If the partitioning number k is given as 3, the maximum capacity δ is given as 4, and the maximum size tolerance γ is given as 2, then 1) the set of eight quantum bits $\{q_1, q_2, q_3, q_4, q_5, q_6, q_7, \text{and } q_8\}$ can be partitioned into three partitions, $\{q_1, q_2, q_5, q_6\}$, $\{q_3, q_7\}$, and $\{q_4, q_8\}$ in the three-way (4, 2)-balanced partitioning and 2) the communication cost can be minimized as 10 in the DQC design. As illustrated in Fig. 1(b), it is clear that the communication in the three-way (4, 2)-balanced partitioning can be placed on the nine global gates U_1, U_2, U_6, U_7, U_{11}, U_{14}, U_{15}, U_{19}, and U_{22}.

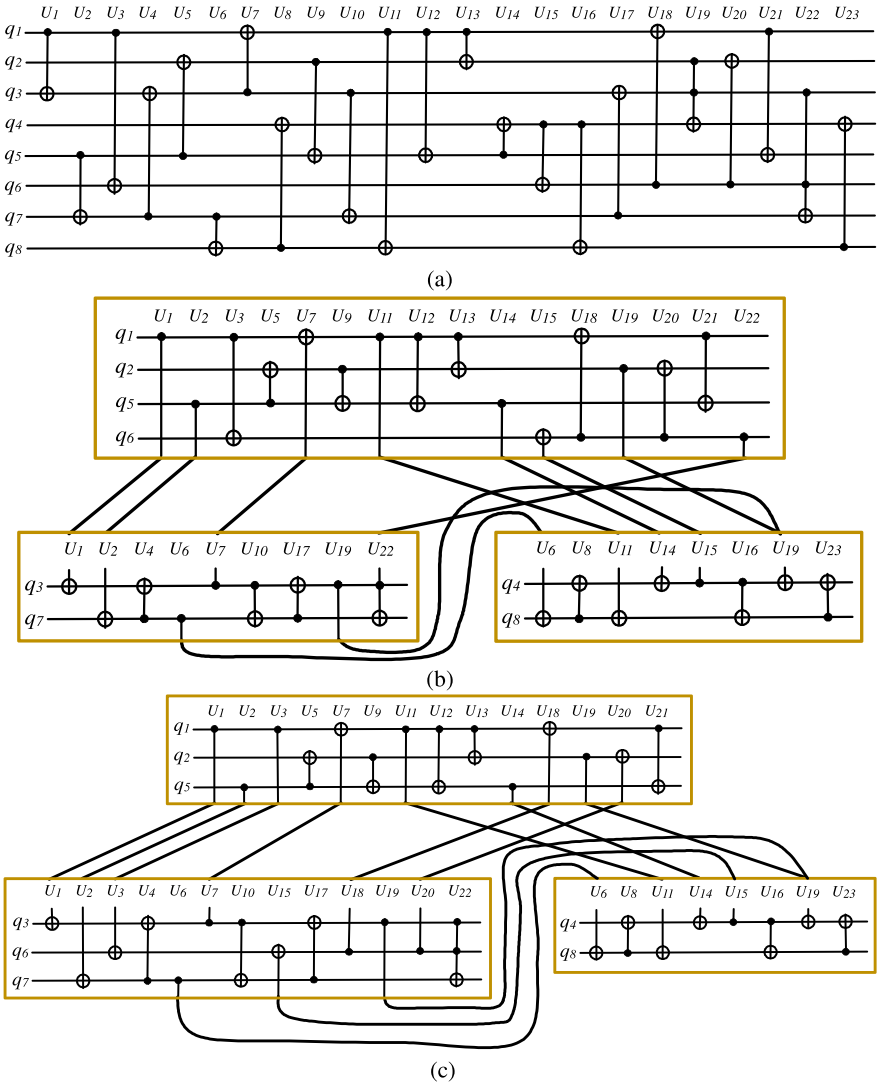


FIGURE 1. **Balanced partitioning for an original QC with 8 qubits and 23 quantum gates in a DQC design. (a) Original QC with 8 qubits and 23 quantum gates. (b) Three-way (4, 2)-balanced partitioning in a DQC design. (c) Three-way (3, 1)-balanced partitioning in a DQC design.**

Similarly, if the partitioning number k is given as 3, the maximum capacity δ is given as 3, and the maximum size tolerance γ is given as 1, then 1) the set of eight quantum bits $\{q_1, q_2, q_3, q_4, q_5, q_6, q_7,$ and $q_8\}$ can be partitioned into three partitions, $\{q_1, q_2, q_5\}$, $\{q_3, q_6, q_7\}$, and $\{q_4, q_8\}$ in the three-way (3, 1)-balanced partitioning and 2) the communication cost can be minimized as 12 in the DQC design. As illustrated in Fig. 1(c), it is clear that the communications in the three-way (3, 1)-balanced partitioning can be placed on the 11 global gates, U_1, U_2, U_3, U_6, U_7, U_{11}, U_{14}, U_{15}, U_8, U_{19}, and U_{20}.

III. BALANCED PARTITIONING UNDER CAPACITY AND SIZE-TOLERANCE CONSTRAINTS IN DQCS

Given an original QC with n qubits and m gates, a partitioning number k, the maximum capacity δ inside each partition, and the maximum size tolerance γ among two partitions, a fuzzy-based partitioning algorithm can be proposed to minimize the communication cost in k-way (δ, γ)-balanced partitioning under the capacity constraint δ and the size-tolerance constraint γ, and the design flow of the proposed algorithm is shown in Fig. 2.

In the proposed algorithm, the process of partitioning an original QC into k partitions in k-way (δ, γ)-balanced partitioning can be divided into three sequential steps: *construction of edge-weighted connection graph, generation of fuzzy matrix in FKGC,* and *vertex assignment in k-way* (δ, γ)*-balanced partitioning.*

For the construction of an edge-weighted connection graph, based on the communication relation inside the gates in a given QC, an edge-weighted connection graph can be constructed. For the generation of a fuzzy matrix in FKGC, first, based on the partitioning number k and the edge connections in the connection graph, the connection strength between two vertices in the connection graph can be estimated. Furthermore, based on the edge connections in the connection graph, the initial k-way partitioning in the connection graph can be constructed by using a bottom-up clustering

IEEE Transactions on Quantum Engineering

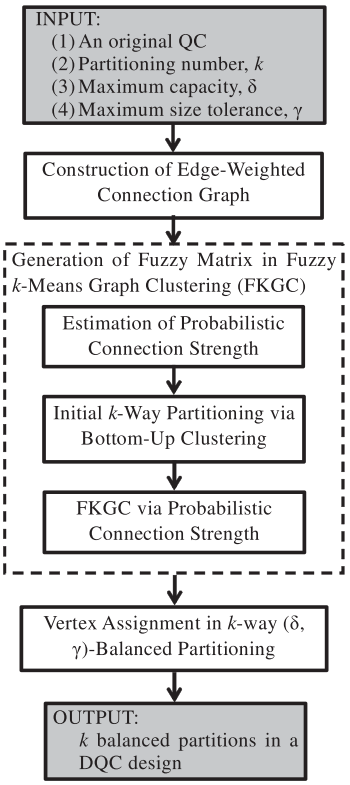


FIGURE 2. Design flow of the proposed partitioning algorithm in *k*-way (δ, γ)-balanced partitioning.

algorithm. Finally, based on the definition of the clustering distance between two vertices in the connection graph, the fuzzy memberships on k clusters can be generated in FKGC. For the vertex assignment in k-way (δ, γ)-balanced partitioning, based on the maximum capacity δ inside each partition, the maximum size tolerance γ between two partitions, and the fuzzy memberships on k clusters, the vertices in the connection graph can be assigned onto k partitions under the capacity constraint δ and the size-tolerance constraint γ.

A. CONSTRUCTION OF EDGE-WEIGHTED CONNECTION GRAPH

Given an original QC with n qubits $q_1, q_2, \ldots, q_n$ and m gates $U_1, U_2, \ldots, U_m$, initially, it is assumed that there is only one operating qubit inside each gate in a given QC. Basically, the communication relation can be defined as the relation between a controlled qubit and its operating qubit inside one gate. It is known that the communication between two qubits is permitted to be bidirectional in a DQC design. Hence, any edge in a connection graph is undirected. In the assignment of a graph edge, the communication cost between two qubits inside a global gate can be set as 1. In contrast, the communication cost between two qubits inside a local gate can be set as 0. Based on any available qubit in a given QC as one vertex and the communication relation between 1-controlled qubit and its operating qubit inside one gate in a given QC

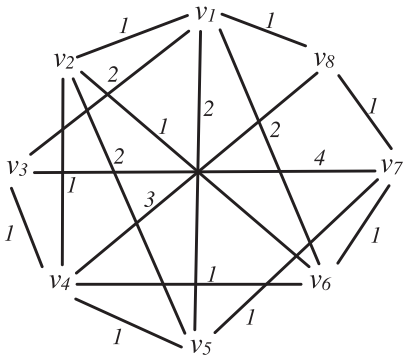


FIGURE 3. Construction of an undirected edge-weighted connection graph for a given QC with 8 qubits and 23 gates.

as one undirected edge, an undirected edge-weighted connection graph $G_c(V_c, E_c)$ can be constructed as follows: a vertex v_j in V_c represents one available qubit q_j, $1 \le j \le n$, and an undirected edge $e_{j,l}$ in E_c represents the communication relation between 1-controlled qubit q_j and its operating qubit q_l inside one gate U_i, $1 \le i \le m$, $1 \le j, l \le n, j \ne l$. In addition, the weight of the undirected edge $e_{j,l}$ in E_c represents the number of gates with the communication relation between the two qubits q_j and q_l.

Refer to the given QC with eight qubits $q_1, q_2, \ldots, q_8$ and 23 gates $U_1, U_2, \ldots, U_{23}$, in Fig. 1(a), based on the eight qubits in the QC as 8 vertices and the 16 communication relations inside 23 gates in the QC as 16 undirected edges, an undirected edge-weighted connection graph $G_c(V_c, E_c)$ can be constructed, where $V_c = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ and $E_c = \{e_{1,2}, e_{1,3}, e_{1,5}, e_{1,6}, e_{1,8}, e_{2,4}, e_{2,5}, e_{2,6}, e_{3,4}, e_{3,7}, e_{4,5}, e_{4,6}, e_{4,8}, e_{5,7}, e_{6,7}, e_{7,8}\}$, as illustrated in Fig. 3. In addition, the weights of the 16 undirected edges can be set as $w(e_{1,2}) = 1$, $w(e_{1,3}) = 2$, $w(e_{1,5}) = 2$, $w(e_{1,6}) = 2$, $w(e_{1,8}) = 1$, $w(e_{2,4}) = 1$, $w(e_{2,5}) = 2$, $w(e_{2,6}) = 1$, $w(e_{3,4}) = 1$, $w(e_{3,7}) = 4$, $w(e_{4,5}) = 1$, $w(e_{4,6}) = 1$, $w(e_{4,8}) = 3$, $w(e_{5,7}) = 1$, $w(e_{6,7}) = 1$, and $w(e_{7,8}) = 1$, respectively.

B. GENERATION OF FUZZY MATRIX IN FKGC

To our knowledge, FKGC on fuzzy c-means clustering has been used for graph bisection and two-way circuit partitioning [23], [24], k-way circuit partitioning [25], partitioning-based placement [26], and layer-aware via minimization [27]. However, the fuzzy matrix in FKGC seriously depends on the definition of the clustering distance between two vertices in a corresponding graph. Due to the definition of the clustering distances in different applications, it is clear that the FKGC in circuit partitioning [23], [24], [25], placement [26], or layer-aware via minimization [27] cannot be directly used in multiple-way balanced partitioning. Hence, it is necessary for the development of multiple-way balanced partitioning in a DQC design to define the accurate clustering distance in FKGC.

For FKGC in multiple-way balanced partitioning, it is known that the construction of an initial k-way partitioning in

the connection graph, the definition of the clustering distance between two vertices in the connection graph, the formulation of one error function, and the selection of an acceptable error ε must be considered. To generate the final fuzzy matrix of a given connection graph in FKGC, the generation process can be further divided into three sequential steps: *estimation of probabilistic connection strength*, *initial k-way partitioning via bottom-up clustering,* and *FKGC via probabilistic connection strength.*

1) ESTIMATION OF PROBABILISTIC CONNECTION STRENGTH

Given an edge-weighted connection graph G_c, there may be some connection paths between any pair of two vertices v_i and v_j. If the vertices v_i and v_j are divided into two different groups, all the connection paths of the vertices v_i and v_j must be fully cut. If one edge on any connection path between the vertices v_i and v_j is cut, the weight sum of all the cut edges between the vertices v_i and v_j can be treated as the connection strength of the vertices v_i and v_j for the cut edges. Since one cut edge on any connection path between the vertices v_i and v_j is randomly selected, all the connection strengths of the vertices v_i and v_j can be obtained for all the possible cut edges. To measure the connection strength of the vertices v_i and v_j, the concept of the probabilistic connection strength of the vertices v_i and v_j can be introduced by using the uniform distribution in probability theory. Hence, the estimation of the probabilistic connection strength between two vertices in the connection graph can be divided into two following steps: *extraction of estimation paths* and *computation of probabilistic connection strength.*

For the extraction of the estimation paths in an edge-weighted connection graph, given any pair of two vertices v_i and v_j in an edge-weighted connection graph G_c, one iterative extraction process can be proposed to find a feasible set of estimation paths between the vertices v_i and v_j in the graph G_c. In the iterative extraction process, first, the maximum-weight shortest path $p_1^{i,j}$ between the vertices v_i and v_j can be found and extracted from the graph G_c as an estimation path. Basically, the vertices, excluding the vertices v_i and v_j, on the path $p_1^{i,j}$ can be defined as a set of *extracted vertices* on the path $p_1^{i,j}$. After extracting the path $p_1^{i,j}$, the extracted vertices and edges on the path $p_1^{i,j}$ and the edges connecting to the extracted vertices must be further deleted from the graph G_c, and the iterative extraction process can continue for the extraction of the next estimation path $p_2^{i,j}$ in the remaining graph G_c. Until there is no path between the vertices v_i and v_j in the remaining graph G_c, the iterative extraction process will stop. As a result, a set of estimation paths between the vertices v_i and v_j can be extracted for the computation of the probabilistic partitioning cut between the vertices v_i and v_j in the graph G_c.

Refer to the two vertices v_1 and v_6 in the connection graph G_c, in Fig. 3, first, the maximum-weight shortest path $p_1^{1,6}$, including the edge $e_{1,6}$, can be extracted from the graph G_c.

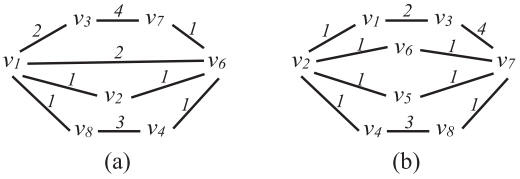


FIGURE 4. Extraction of estimation paths in an edge-weighted connection graph. (a) Extraction of one direct path and three indirect paths between two vertices v_1 and v_6. (b) Extraction of four indirect paths between two vertices v_2 and v_7.

After extracting the path $p_1^{1,6}$, the edge $e_{1,6}$ must be deleted from the graph G_c. Furthermore, the maximum-weight shortest path $p_2^{1,6}$, including the two edges $e_{1,2}$ and $e_{2,6}$, can be extracted from the remaining graph G_c. After extracting the path $p_2^{1,6}$, the vertex v_2, the two edges $e_{1,2}$ and $e_{2,6}$, and the two edges $e_{2,4}$ and $e_{2,5}$, connecting to the vertex v_2, must be deleted from the remaining graph G_c. Next, the maximum-weight shortest path $p_3^{1,6}$, including the three edges $e_{1,3}$, $e_{3,7}$, and $e_{6,7}$, can be extracted from the remaining graph G_c. After extracting the path $p_3^{1,6}$, the two vertices v_3 and v_7, the three edges $e_{1,3}$, $e_{3,7}$, and $e_{6,7}$, the edge $e_{3,4}$ connecting to the vertex v_3, and the two edges $e_{5,7}$ and $e_{7,8}$ connecting to the vertex v_7 must be deleted from the remaining graph G_c. Finally, the maximum-weight shortest path $p_4^{1,6}$, including the three edges $e_{1,8}$, $e_{4,6}$, and $e_{4,8}$, can be extracted from the remaining graph G_c. After extracting the path $p_4^{1,6}$, the two vertices v_4 and v_8, the three edges $e_{1,8}$, $e_{4,6}$, and $e_{4,8}$, and the edge $e_{4,5}$ connecting to the vertex v_4 must be deleted from the remaining graph G_c. As a result, the four estimation paths $p_1^{1,6}$, $p_2^{1,6}$, $p_3^{1,6}$, and $p_4^{1,6}$ between the vertices v_1 and v_6 can be extracted from the graph G_c, as illustrated in Fig. 4(a).

Similarly, refer to the two vertices v_2 and v_7 in the connection graph G_c, in Fig. 3, first, the maximum-weight shortest path $p_1^{2,7}$, including the two edges $e_{2,5}$ and $e_{5,7}$, can be extracted from the graph G_c. After extracting the path $p_1^{2,7}$, the vertex v_5, the two edges $e_{2,5}$ and $e_{5,7}$, and the two edges $e_{1,5}$ and $e_{4,5}$ connecting to the vertex v_5 must be deleted from the graph G_c. Furthermore, the maximum-weight shortest path $p_2^{2,7}$, including the two edges $e_{2,6}$ and $e_{6,7}$, can be extracted from the remaining graph G_c. After extracting the path $p_2^{2,7}$, the vertex v_6, the two edges $e_{2,6}$ and $e_{6,7}$, and the two edges $e_{1,6}$ and $e_{4,6}$ connecting to the vertex v_6 must be deleted from the remaining graph G_c. Next, the maximum-weight shortest path $p_3^{2,7}$, including the three edges $e_{1,2}$, $e_{1,3}$, and $e_{3,7}$, can be extracted from the remaining graph G_c. After extracting the path $p_3^{2,7}$, the two vertices v_1 and v_3, the three edges $e_{1,2}$, $e_{1,3}$, and $e_{3,7}$, the three edges $e_{1,5}$, $e_{1,6}$, and $e_{1,8}$ connecting to the vertex v_1, and the edge $e_{3,4}$ connecting to the vertex v_3 must be deleted from the remaining graph G_c. Finally, the maximum-weight shortest path $p_4^{2,7}$, including the three edges $e_{2,4}$, $e_{4,8}$, and $e_{7,8}$, can be extracted from the remaining graph G_c. After extracting the path $p_4^{2,7}$, the two vertices v_4 and v_8 and the three edges $e_{2,4}$, $e_{4,8}$, and $e_{7,8}$ must

IEEE Transactions on Quantum Engineering

be deleted from the remaining graph G_c. As a result, the four estimation paths $p_1^{2,7}$, $p_2^{2,7}$, $p_3^{2,7}$, and $p_4^{2,7}$ between the vertices v_2 and v_7 can be extracted from the graph G_c, as illustrated in Fig. 4(b).

For the computation of the probabilistic connection strength between the vertices v_i and v_j in an edge-weighted connection graph G_c given a set of r estimation paths $p_1^{i,j}$, $p_2^{i,j}$, …, $p_r^{i,j}$ between the vertices v_i and v_j, it is assumed that there are n_h edges on the hth estimation path $p_h^{i,j}$, and there are n_h weights $w_{h,1}, w_{h,2}, \ldots, w_{h\,nh}$ on the n_h edges, $1 \le h \le r$. If there is no estimation path between the vertices v_i and v_j, the probabilistic partitioning cut $\text{ppc}_{i,j}$ between the vertices v_i and v_j can be set as 0. On the other hand, if the vertex v_i is the same as the vertex v_j, the probabilistic partitioning cut $\text{ppc}_{i,j}$ between the vertices v_i and v_j can be set as ∞. Based on the uniform distribution in probability theory, the probabilistic partitioning cut $\text{ppc}_{i,j}$ between the vertices v_i and v_j on the r estimation paths $p_1^{i,j}$, $p_2^{i,j}$, …, $p_r^{i,j}$ can be further computed and set as

$$\text{ppc}_{i,j} = \frac{\sum_{h=1}^{r}\left[\prod_{s=1,s\neq h}^{r} n_s \sum_{t=1}^{n_h} w_{h,t}\right]}{\prod_{h=1}^{r} n_h}, \quad \text{if } r > 0 \text{ and } v_i \neq v_j$$
$$= 0, \quad \text{if } r = 0 \text{ and } v_i \neq v_j$$
$$= \infty, \quad \text{if } v_i = v_j.$$

As a result, the matrix M_{PPC} representing the probabilistic partitioning cuts in the graph G_c can be obtained.

It is known that the larger the probabilistic partitioning cut between two vertices is, the larger the connection strength between two vertices is. To estimate the probabilistic connection strength between the vertices v_i and v_j given the upper bound of the maximum cut U_{MC} as the weight sum of all the edges in the graph G_c, the probabilistic connection strength $\text{pcs}_{i,j}$ between the vertices v_i and v_j in the graph G_c can be defined as the ration between the probabilistic partitioning cut between the vertices v_i and v_j, and the upper bound of the maximum cut in the graph G_c. If the vertex v_i is the same as the vertex v_j, the probabilistic connection strength $\text{pcs}_{i,j}$ between the vertices v_i and v_j can be set as 1. If the vertex v_i is different from the vertex v_j, the probabilistic connection strength $\text{pcs}_{i,j}$ between the vertices v_i and v_j can be set as $\text{ppc}_{i,j}/U_{MC}$. As a result, the matrix M_{PCS}, representing the probabilistic connection strength in the graph G_c, can be obtained.

Refer to the edge-weighted connection graph G_c in Fig. 3, based on the extraction of the estimation paths between two vertices in the graph G_c, the matrix M_{PPC}, representing the probabilistic partitioning cuts in the graph G_c, can be obtained, as illustrated in Fig. 5(a). Based on the edge weights in the graph G_c, the upper bound of the maximum cut in the graph G_c can be obtained as 23. Furthermore, based on the matrix representing the probabilistic partitioning cuts in the graph G_c, the matrix M_{PCS}, representing the probabilistic connection strength in the graph G_c, can be obtained, as illustrated in Fig. 5(b).

$$M_{PPC} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{array} \begin{array}{c} \begin{array}{cccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \end{array} \\ \begin{bmatrix} \infty & 6.17 & 6.00 & 7.50 & 7.50 & 7.00 & 7.00 & 5.33 \\ 6.17 & \infty & 4.83 & 5.17 & 5.50 & 4.83 & 6.83 & 4.33 \\ 6.00 & 4.83 & \infty & 5.00 & 5.50 & 5.50 & 7.33 & 6.00 \\ 7.50 & 5.17 & 5.00 & \infty & 6.50 & 5.67 & 6.50 & 6.33 \\ 7.50 & 5.50 & 5.50 & 6.50 & \infty & 5.50 & 6.67 & 4.50 \\ 7.00 & 4.83 & 5.50 & 5.67 & 5.50 & \infty & 6.67 & 4.50 \\ 7.00 & 6.83 & 7.33 & 6.50 & 6.67 & 6.67 & \infty & 5.00 \\ 5.33 & 4.33 & 6.00 & 6.33 & 4.50 & 4.50 & 5.00 & \infty \end{bmatrix} \end{array}$$

(a)

$$M_{PCS} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{array} \begin{array}{c} \begin{array}{cccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \end{array} \\ \begin{bmatrix} 1 & 0.268 & 0.261 & 0.326 & 0.326 & 0.304 & 0.304 & 0.232 \\ 0.268 & 1 & 0.210 & 0.225 & 0.239 & 0.210 & 0.297 & 0.188 \\ 0.261 & 0.210 & 1 & 0.217 & 0.239 & 0.239 & 0.319 & 0.261 \\ 0.326 & 0.225 & 0.217 & 1 & 0.283 & 0.247 & 0.283 & 0.275 \\ 0.326 & 0.239 & 0.239 & 0.283 & 1 & 0.239 & 0.290 & 0.196 \\ 0.304 & 0.210 & 0.239 & 0.247 & 0.239 & 1 & 0.290 & 0.196 \\ 0.304 & 0.297 & 0.319 & 0.283 & 0.290 & 0.290 & 1 & 0.217 \\ 0.232 & 0.188 & 0.261 & 0.275 & 0.196 & 0.196 & 0.217 & 1 \end{bmatrix} \end{array}$$

(b)

FIGURE 5. Estimation of probabilistic connection strength in an edge-weighted connection graph. (a) Matrix M_{PPC} representing probabilistic partitioning cuts in an edge-weighted connection graph. (b) Matrix M_{PCS} representing probabilistic connection strength in an edge-weighted connection graph.

2) INITIAL *K*-WAY PARTITIONING VIA BOTTOM-UP CLUSTERING

Given an edge-weighted connection graph G_c (V_c, E_c), $V_c = \{v_1, v_2, \ldots, v_n\}$ and a partitioning number k, *k-way graph partitioning* (KGP) can be defined by specifying the vertices in V_c into k subsets $v_1, v_2, \ldots, V_k$. Furthermore, the KGP result can be represented by using k characteristic functions u_i: V_c –> {0, 1} for the ith vertex subset V_i, $1 \le i \le k$, as follows:

$$u_i(v_j) = \begin{cases} 0, & \text{if } v_j \notin V_i \\ 1, & \text{if } v_j \in V_i. \end{cases}$$

As a result, a partitioning matrix can be used to represent the k characteristic functions for the result of the k partitions in the graph G_c.

It is known that a better initial KGP result can reduce the convergence time in FKGC. In FKGC, an initial KGP result in the graph G_c can be constructed by using one iterative bottom-up clustering process. Initially, each vertex v_j in the graph G_c can be treated as one partition P_j, $1 \le j \le n$. If the partition number is larger than k in the graph G_c, the edge $e_{i,j}$ with the largest weight $w(e_{i,j})$ in the graph G_c can be selected and the two partitions P_i and P_j connected by using the edge $e_{i,j}$ can be merged into one larger partition $P_{(i,j)}$. After constructing the new partition $P_{(i,j)}$, the graph G_c must

Algorithm 1: KGPBUC.

Input: An edge-weighted connection graph G_c with n vertices;
A partitioning number k;
Output: A set of k subgraphs $G_1, G_2, \ldots, G_k$ for k partitions;

Set an initial set of n subgraphs for n initial partitions P_1, P_2, and P_n, where $P_i = \{v_i\}$, $1 \le i \le n$;
while (Partition number in G_c is larger than k)
  Find one edge $e_{i,j}$ with the largest weight between two partitions P_i and P_j in G_c;
  Merge the two partitions P_i and P_j into one larger partition $P_{(i,j)}$;
  Modify the graph G_c by using the partition $P_{(i,j)}$ as one new vertex and summing the weights on the corresponding merged edges;
end while
return A set of k subgraphs $G_1, G_2, \ldots, G_k$ for k partitions;

be modified by using the partition $P_{(i,j)}$ and summing the weights on the corresponding merged edges. Furthermore, the iterative clustering process will continue for the modified graph G_c. Until the partition number is equal to k in the modified graph G_c, the iterative clustering process will stop. As a result, a partitioning matrix M_0 can be used to represent the result of the k partitions in the graph G_c for the initial KGP result using in FKGC.

Given an edge-weighted connection graph G_c and a partitioning number k, the KGP result can be obtained by running the iterative bottom-up clustering algorithm, k-way graph partitioning via bottom-up clustering (KGPBUC).

Refer to the edge-weighted connection graph G_c in Fig. 3, if the partitioning number is given as 3, one iterative bottom-up clustering process can be used to construct three partitions in the graph G_c. Initially, the eight vertices in the graph G_c can be set as eight initial partitions. As illustrated in Fig. 6(a), in the first iteration, the vertices v_3 and v_7 representing the two partitions P_3 and P_7 can be merged into one new vertex (v_3, v_7), representing the merged partition $P_{(3, 7)}$ in the modified graph G_c. In the second iteration, the vertices v_4 and v_8, representing the two partitions P_4 and P_8, can be merged into one new vertex (v_4, v_8), representing the merged partition $P_{(4, 8)}$, in the modified graph G_c. In the third iteration, the vertices v_1 and (v_3, v_7), representing the two partitions P_1 and $P_{(3, 7)}$, can be merged into one new vertex (v_1, v_3, v_7), representing the merged partition $P_{(1, 3, 7)}$ in the modified graph G_c. In the fourth iteration, the vertices v_6 and (v_1, v_3, v_7), representing the two partitions P_6 and $P_{(1, 3, 7)}$, can be merged into one new vertex (v_1, v_3, v_6, v_7), representing the merged partition $P_{(1, 3, 6, 7)}$ in the modified graph G_c. In the fifth iteration, the vertices (v_4, v_8) and (v_1, v_3, v_6, v_7), representing the two partitions $P_{(4, 8)}$ and $P_{(1, 3, 6, 7)}$, can be merged into one new vertex $(v_1, v_3, v_4, v_6, v_7, v_8)$,

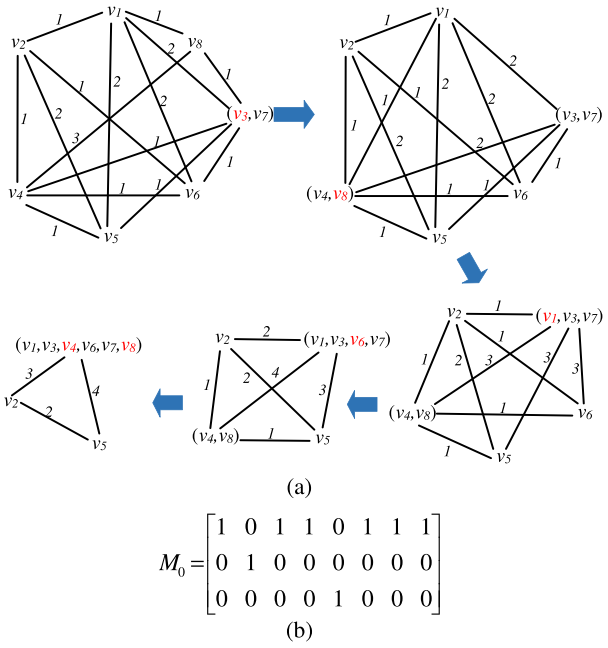


FIGURE 6. Construction of three-way graph partitioning in an edge-weighted connection graph. (a) Three-way graph partitioning via iterative bottom-up clustering. (b) Partitioning matrix for three-way graph partitioning result.

representing the merged partition $P_{(1, 3, 4, 6, 7, 8)}$ in the modified graph G_c. As a result, the eight vertices in the graph G_c can be partitioned into three partitions $\{v_1, v_3, v_4, v_6, v_7, v_8\}$, $\{v_2\}$, and $\{v_5\}$, and the partitioning cut can be obtained as 9 in three-way graph partitioning. After completing the iterative bottom-up clustering process, the partitioning matrix M_0, representing the result of the three partitions $\{v_1, v_3, v_4, v_6, v_7, v_8\}$, $\{v_2\}$, and $\{v_5\}$, in the graph G_c can be used as an initial partitioning result using in FKGC, as illustrated in Fig. 6(b).

3) FKGC VIA PROBABILISTIC CONNECTION STRENGTH

Given an edge-weighted connection graph G_c, the probabilistic connection strength $\text{pcs}_{i,j}$ between two vertices v_i and v_j, $1 \le i, j \le n$, in the graph G_c can be computed. Clearly, the probabilistic connection strength $\text{pcs}_{i,j}$ between two vertices v_i and v_j can further reflect the clustering distance $d_{i,j}$ between two vertices v_i and v_j in the graph G_c. The larger the probabilistic connection strength $\text{pcs}_{i,j}$ between two vertices v_i and v_j in the graph G_c is, the shorter the clustering distance between two vertices v_i and v_j in the graph G_c is. Based on the probabilistic connection strength $\text{pcs}_{i,j}$ between two vertices v_i and v_j, the clustering distance $d_{i,j}$ between two vertices v_i and v_j in the graph G_c can be defined as follows:

$$d_{i,j} = \begin{cases} \infty, & \text{if } \text{pcs}_{i,j} = 0 \\ \frac{1}{\text{pcs}_{i,j}}, & \text{if } 0 < \text{pcs}_{i,j} < 1 \\ 0, & \text{if } \text{pcs}_{i,j} = 1. \end{cases}$$

Based on the concept of fuzzy c-means clustering on the geometrical clustering distances, the FKGC in the graph G_c

on the defined clustering distance $d_{i,j}$ between two vertices v_i and v_j, $1 \le i, j \le n$, can be designed as follows. Similar to the definition of the characteristic function for the ith vertex subset V_i, $1 \le i \le k$, in KGP, the characteristic function can be defined as a fuzzy function u_i: V_c –> [0, 1] for the ith vertex set V_i, $1 \le i \le k$, in FKGC. In KGP, the restricted assignment of any vertex in the graph G_c onto one partition may lead to a nonbalanced partitioning result. In contrast, each vertex in the graph G_c can be specified by using the degree of belonging to each partition in FKGC and the concept of using the fuzzy memberships in FKGC can easily lead to satisfy some specific size constraints. If there are k fuzzy functions $u_1, u_2, \ldots, u_k$ associated with the vertex set V_c in the graph G_c, the partitioning result of each vertex v_i in the vertex set V_c can be represented by using the k fuzzy memberships $u_{1,i}, u_{2,i}, \ldots, u_{k,i}$ onto k clusters in FKGC. Hence, the main purpose of partitioning the vertex set V_c in FKGC is to find the k fuzzy memberships of each vertex in the vertex set V_c onto k clusters.

Given the vertex set $V_c = \{v_1, v_2, \ldots, v_n\}$ in the graph G_c, the k fuzzy functions of the vertices in the vertex set V_c can be represented by a fuzzy matrix $U \in M_{fkm}$, where M_{fkm} is all possible fuzzy matrices in FKGC. In the formulation of one error function, given the clustering distances $d_{i,j}$ between two vertices v_i and v_j, $1 \le i, j \le n$, in the graph G_c, the classical squared-error function [28] can be formulated and used in FKGC as follows:

$$J(U, vc, G_c) = \sum_{j=1}^{n} \sum_{i=1}^{k} (u_{i,j})^2 (d_{c(i),j})^2$$

$$\text{subject to } \sum_{i=1}^{k} u_{i,j} = 1$$

where $vc_i = v_{c(i)} \in V_c$, $1 \le i \le k$, is the center of the ith cluster.

Basically, the objective function J can be treated as a squared-error criterion and its minimization can further produce a fuzzy matrix U that is optimal in a least squared error. To approximately minimize the objective function J, the objective function J can be minimized on the fuzzy memberships $u_{i,j}$, $1 \le i \le k$, $1 \le j \le n$, and the centers vc_i, $1 \le i \le k$, by using an iterative improvement algorithm. In the iterative improvement algorithm, based on the optimality analysis of the fuzzy c-means clustering [29], [30] on the geometrical clustering distances, the optimization of the objective function J under some constraints can be obtained by alternately finding the partial optimization for the modification of the fuzzy memberships $u_{i,j}$, $1 \le i \le k$, $1 \le j \le n$, and the optimization for the selection of the k centers vc_i, $1 \le i \le k$. In the modification of the fuzzy memberships $u_{i,j}$, $1 \le i \le k$, $1 \le j \le n$, the center vector vc in the function J must be fixed and the necessary condition of the fuzzy matrix $M = \{u_{i,j}\}$ can be found by the Lagrange multiplier method to minimize the function J. In the selection of the k centers vc_i, $1 \le i \le k$, the fuzzy matrix $M = \{u_{i,j}\}$ in the function J must be fixed and the necessary condition of the center vector vc can be found by an exhaustive search to minimize the function J. As a result, a final fuzzy k-means matrix M can be used to represent the fuzzy membership of the k partitions in the graph G_c.

TABLE 1. Symbols and Notations in Iterative Improvement Algorithm ε-FKGC

Symbols	Definitions
k	A partitioning number
ε	An acceptable error
$pcs_{i,j}$	probabilistic connection strength between two vertices, v_i and v_j, in G_c
$d_{i,j}$	clustering distance between two vertices, v_i and v_j, in G_c
vc_i	i-th center inside the k clusters
$u_{i,j}$	the fuzzy membership of the j-th vertex onto *the* i-th cluster in a fuzzy matrix

Table 1 presents the symbols and notations in the iterative improvement algorithm ε-FKGC.

Based on the definition of the clustering distance $d_{i,j}$ between two vertices v_i and v_j, $1 \le i, j \le n$, in the graph G_c, the KGP matrix M_0, representing the k-way partitioning result of all the vertices in the graph G_c as an initial partitioning result, the selection of an acceptable error ε, the necessary condition of the fuzzy matrix M, the center vector vc, and a ε-approximate fuzzy matrix M in FKGC can be obtained by using the iterative improvement algorithm ε-FKGC on the fuzzy matrix M and the cluster centers vc for the objective function J.

Refer to the matrix M_{PCS}, representing the probabilistic connection strength in the graph G_c, in Fig. 5(b) and the KGP matrix M_0 for three-way graph partitioning in Fig. 6(b), based on the definition of the clustering distance $d_{i,j}$ between two vertices v_i and v_j, $1 \le i, j \le n$, in the graph G_c, the clustering distance $d_{i,j}$ between two vertices v_i and v_j, $1 \le i, j \le n$, in the graph G_c can be computed. By using the KGP matrix M_0, as the initial fuzzy matrix and the iterative improvement on the modification of the fuzzy memberships and the selection of the three centers in fuzzy 3-means graph clustering, as illustrated in Fig. 7(a), the final ε-approximate fuzzy matrix M of the fuzzy 3-means graph clustering can be obtained after completing the algorithm ε-FKGC with $\varepsilon = 0.001$, as illustrated in Fig. 7(b).

C. VERTEX ASSIGNMENT IN K-WAY (δ, γ)-BALANCED PARTITIONING

For the vertex assignment in k-way (δ, γ)-balanced partitioning, based on the fuzzy memberships on the n vertices inside k clusters in a final fuzzy matrix $M = \{u_{i,j}\}$, $1 \le i \le k$, $1 \le j \le n$, the maximum capacity δ, and the maximum size tolerance γ, the vertex assignment in k-way (δ, γ)-balanced partitioning can be divided into two sequential steps: *Initial assignment* and *Iterative size modification*.

In the initial assignment of all the vertices, the assignment of the n vertices v_1, v_2, …, and v_n in the graph G_c is based on the largest fuzzy membership $u_{i,j}$, $1 \le i \le k$,

Algorithm 2: ε-FKGC.

Input: An edge-weighted connection graph $G_c(V_c, E_c)$;
A matrix of probabilistic connection strength $pcs_{i,j}$ between two vertices v_i and v_j, $1 \le i, j \le n$, in G_c;
A partitioning number k;
An acceptable error ε;
Output: A fuzzy matrix M of all the vertices in G_c on k clusters;

Determine the clustering distance $d_{i,j}$ between two vertices v_i and v_j based on the probabilistic connection strength $pcs_{i,j}$, $1 \le i, j \le n$, in G_c;
Run the KGPBUC for the vertices in G_c and establish the initial fuzzy matrix M_0;
$M^* = M_0$;
do
  $M = M^*$;
  Find the centers set $vc = (vc_1, vc_2, \ldots, vc_k)$ of the k clusters using the fuzzy membership $u_{i,j}$ inside M and the clustering distance $d_{i,j}$ as

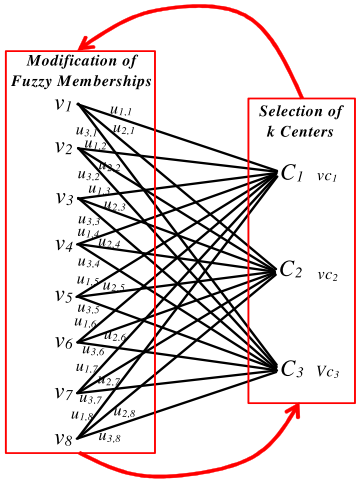
$$vc_i = v_{c(i)}, \text{MIN} \sum_{j=1}^{n} (u_{i,j})^2 (d_{c(i),j})^2, v_{c(i)} \in V_c;$$

  for $j = 1$ **to** n
    for $i = 1$ **to** k
      if $v_j \notin \{vc_1, vc_2, \ldots, vc_k\}$
        Calculate the fuzzy membership $u^*_{i,j}$ of the vertex v_j on the ith cluster in M^* using the centers $vc = (vc_1, vc_2, \ldots, vc_k)$ as

$$u^*_{i,j} = \frac{1}{\sum_{p=1}^{k} \left(\frac{d_{c(i),j}}{d_{c(p),j}} \right)^2};$$

      else
        if $v_j = vc_i$
          $u^*_{i,j} = 1$;
        else
          $u^*_{i,j} = 0$;
        end if
      end if
    end for
  end for
  Compare $M = \{u_{i,j}\}$ and $M^* = \{u^*_{i,j}\}$;
until $|u^*_{i,j} - u_{i,j}| < \varepsilon$, for $1 \le i \le k$, $1 \le j \le n$;
$M = M^*$;
return A fuzzy matrix M of all the vertices in G_c on k clusters;

$1 \le j \le n$, on the k clusters. If the larger fuzzy membership of the vertex v_j, $1 \le j \le n$, in the graph G_c is the fuzzy membership $u_{i,j}$ on the ith cluster, $1 \le i \le k$, the vertex v_j can be directly assigned onto the ith cluster. As a result, the n vertices $v_1, v_2, \ldots, v_n$ in the graph G_c can be partitioned into the k partitions $P_1, P_2, \ldots, P_k$. If the number of vertices inside each partition satisfies the given capacity constraint δ and the maximum size difference between two partitions satisfies the given size-tolerance constraint γ, then the initial



(a)

$$M = \begin{bmatrix} 1 & 0.369 & 0.328 & 0 & 0.363 & 0.361 & 0 & 0.312 \\ 0 & 0.336 & 0.398 & 0 & 0.322 & 0.345 & 1 & 0.296 \\ 0 & 0.295 & 0.274 & 1 & 0.315 & 0.294 & 0 & 0.392 \end{bmatrix}$$

(b)

FIGURE 7. Fuzzy 3-means graph clustering in an edge-weighted connection graph. (a) Iterative improvement on modification of fuzzy memberships and selection of three centers in fuzzy 3-means graph clustering. (b) Final fuzzy 3-means matrix.

k-way partitioning result can be obtained as the final k-way (δ, γ)-balanced partitioning result and the partitioning cut in k-way (δ, γ)-balanced partitioning can be computed. On the other hand, if the maximum size inside any partition does not satisfy the given capacity constraint δ or the maximum size difference between two partitions does not satisfy the size-tolerance constraint γ, then the iterative size modification must be used for the k partitions $P_1, P_2, \ldots, P_k$ in the initial assignment.

In the iterative size modification for the k partitions $P_1, P_2, \ldots, P_k$, the iterative size modification inside the k partitions $P_1, P_2, \ldots, P_k$ is based on the reassignment of the vertices from some larger partitions to the smallest partition. Initially, each vertex v_j, $1 \le j \le n$, in the graph G_c can be set as one *original vertex* inside its partition. In each iteration, first, the qth partition P_q with the smallest size can be found from the k partitions $P_1, P_2, \ldots, P_k$, and the original vertices inside the other larger partitions can be treated as the selected vertices in the modification process. For any possible selected vertex v_j inside the ith partition P_i, the membership difference of the vertex v_j in the modification process can be computed and obtained as $(u_{i,j} - u_{q,j})$. Furthermore, the selected vertex v_j with the smallest membership difference can be reassigned into the qth partition P_q, and the selected vertex v_j can be set as one assigned vertex in the modification process. Until the number of vertices inside each partition satisfies the given capacity constraint δ and the maximum size difference between two partitions satisfies the given size-tolerance constraint γ, the iterative size modification for the k partitions $P_1, P_2, \ldots, P_k$ will stop.

IEEE Transactions on Quantum Engineering

Algorithm 3: KBP

Input: A fuzzy matrix $M = \{u_{i,j}\}$, $1 \le i \le k$, $1 \le j \le n$, with fuzzy memberships $u_{i,j}$ on k clusters for an edge-weighted graph G_c;
Maximum capacity δ inside each partition;
Maximum size tolerance γ among two partitions;
Output: k sets of vertices $V_1, V_2, \ldots, V_k$ for k partitions $P_1, P_2, \ldots, P_k$, and the partitioning cut in k-way (δ, γ)-balanced partitioning;

Initialize k sets of vertices, $V_1 = V_2 = \ldots = V_k = \emptyset$;
for $j = 1$ **to** n
  if (the larger fuzzy membership of the vertex v_j is the fuzzy membership $u_{i,j}$ on the ith cluster) **then**
    $V_i = V_i \cup \{v_j\}$;
  end if
end for
if (the number of vertices inside any partition is larger than δ or the size difference between any pair of two partitions is larger than γ) **then**
    Set each vertex in G_c as one original vertex inside its partition;
    while (the maximum size inside any partition is larger than δ or the maximum size difference between two partitions is larger than γ) **then**
      Find the qth partition P_q with the smallest size;
      Set the original vertices inside the other larger partitions as the selected vertices;
      Find the selected vertex v_j with the smallest membership difference from the partition P_i to the partition P_q;
      $V_q = V_q \cup \{v_j\}$;
      $V_j = V_j - \{v_j\}$;
      Set the selected vertex v_j as one reassigned vertex;
    end while
end if
Set the k partitions $P_1, P_2 \ldots, P_k$ as the k-way (δ, γ)-balanced partitioning result;
Compute the partitioning cut in k-way (δ, γ)-balanced partitioning;
return k sets of vertices $V_1, V_2, \ldots, V_k$ for k partitions $P_1, P_2, \ldots, P_k$ and the partitioning cut in k-way (δ, γ)-balanced partitioning;

As a result, the k-way partitioning result can be obtained as the k-way (δ, γ)-balanced partitioning result and the partitioning cut in k-way (δ, γ)-balanced partitioning can be computed.

Based on the fuzzy memberships $u_{i,j}$, $1 \le i \le k$, $1 \le j \le n$, of all the n vertices $v_1, v_2, \ldots, v_n$ on the k clusters in a final ε-approximate fuzzy matrix $M = \{u_{i,j}\}$, $1 \le i \le k$, $1 \le j \le n$, the maximum capacity δ, and the maximum size tolerance γ, the k-way (δ, γ)-balanced partitioning result can be obtained by using the algorithm KBP.

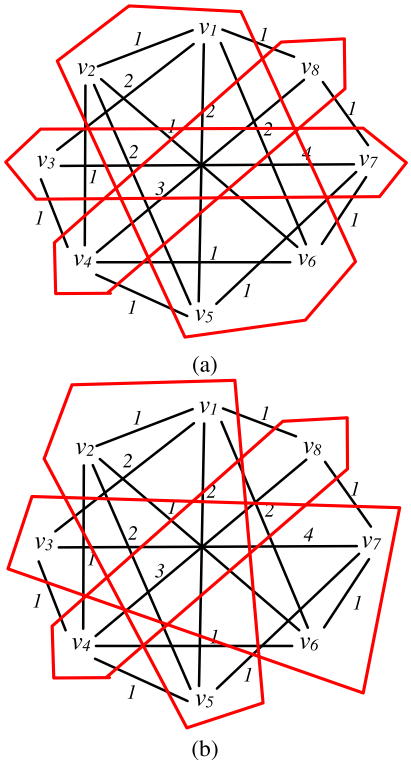


FIGURE 8. Three-way balanced partitioning in an edge-weighted connection graph for an original QC with 8 qubits and 23 gates. (a) Three-way (4, 2)-balanced partitioning result. (b) Three-way (3, 1)-balanced partitioning result.

Refer to the final fuzzy matrix M for fuzzy 3-means graph clustering in Fig. 7(b), based on the fuzzy memberships of all the eight vertices v_1, v_2, v_3, v_4, v_5, v_6, v_7, and v_8 on the three clusters in the final fuzzy matrix M, the three partitions P_1, P_2, and P_3 of the eight vertices can be obtained as $\{v_1, v_2, v_5, v_6\}$, $\{v_3, v_7\}$, and $\{v_4, v_8\}$ in the initial assignment. If the maximum capacity δ is given as 4 and the maximum size tolerance γ is given as 2, it is clear that the 3-way partitioning result can satisfy the capacity constraint $\delta = 4$ and the size-tolerance constraint $\gamma = 2$ in 3-way (4, 2)-balanced partitioning. Based on the three partitions, $P_1 = \{v_1, v_2, v_5, v_6\}$, $P_2 = \{v_3, v_7\}$, and $P_3 = \{v_4, v_8\}$ of the eight vertices surrounded by three red regions, as illustrated in Fig. 8(a), it is clear that a set of eight quantum bits $\{q_1, q_2, q_3, q_4, q_5, q_6, q_7, \text{and } q_8\}$ can be partitioned into three subsets $\{q_1, q_2, q_5, q_6\}$, $\{q_3, q_7\}$, and $\{q_4, q_8\}$, and the communication cost in 3-way (4, 2)-balanced partitioning can be obtained as 10 in the DQC design.

On the other hand, if the maximum capacity δ is given as 3 and the maximum size tolerance γ is given as 1, the initial three partitions $P_1 = \{v_1, v_2, v_5, v_6\}$, $P_2 = \{v_3, v_7\}$, and $P_3 = \{v_4, v_8\}$ of the eight vertices cannot satisfy the capacity constraint $\delta = 3$ and the size-tolerance constraint $\gamma = 1$. Hence, the iterative size modification must be used for the initial 3-way partitioning result.

In the iterative size modification, the smallest partition P_2 can be selected and the four vertices v_2, v_2, v_5, and v_6 inside

the partition P_1 can be treated as the selected vertices in the first iteration. In the iteration, the membership differences of the four vertices v_1, v_2, v_5, and v_6 from the partition P_1 to the partition P_2 can be obtained as 1, 0.023, 0.041, and 0.016, respectively. Hence, the vertex v_6 with the smallest membership difference can be reassigned onto the partition P_2, satisfying the capacity constraint δ=3 and the size-tolerance constraint $\gamma = 1$ in 3-way (3, 1)-balanced partitioning. Based on the three partitions, $P_1 = \{v_1, v_2, v_5\}$, $P_2 = \{v_3, v_6, v_7\}$, and $P_3 = \{v_4, v_8\}$ of the eight vertices surrounded by three red regions, as illustrated in Fig. 8(b), it is clear that a set of eight quantum bits $\{q_1, q_2, q_3, q_4, q_5, q_6, q_7$, and $q_8\}$ can be partitioned into three subsets $\{q_1, q_2, q_5\}$, $\{q_3, q_6, q_7\}$, and $\{q_4, q_8\}$, and the communication cost in 3-way (3, 1)-balanced partitioning can be obtained as 12 in the DQC design.

D. ANALYSIS OF TIME COMPLEXITY

In k-way (δ, γ)-balanced partitioning, the process of partitioning a given QC into k partitions in a DQC design can be divided into three sequential steps: *construction of edge-weighted connection graph, generation of fuzzy matrix in FKGC,* and *vertex assignment in k-way* (δ, γ)*-balanced partitioning.*

For the construction of an edge-weighted connection graph, based on the communication relation inside each gate by scanning all the gates in a given QC, the time complexity of constructing an edge-weighted connection graph is $O(n+m)$, where n is the number of qubits and m is the number of gates in a given QC.

For the generation of a fuzzy matrix in FKGC, first, the time complexity of constructing the estimation paths between two vertices in an edge-weighted connection graph by using the maximum-weight shortest-path algorithm is $O(n^4)$ and the time complexity of computing the probabilistic connection strength between two vertices in an edge-weighted connection graph is $O(n^4)$. Furthermore, the time complexity of constructing the initial KGPBUC in the connection graph is $O(n^2)$. Next, based on the matrix representing the probabilistic connection strength in the connection graph, the time complexity of computing the clustering distance between two vertices in the connection graph is $O(n^2)$. Finally, given a tolerant error ε on the fuzzy memberships, the ε-approximate fuzzy matrix M can be obtained by running the algorithm ε-FKGC. In the iterative improvement of the fuzzy memberships in the generation of the ε-approximate fuzzy matrix M, the number of iterations depends seriously on the number of vertices n in the connection graph and the tolerant error ε. Clearly, the more the number of vertices or the smaller the tolerant error, ε is, the more the number of iterations is. However, the number of iterations cannot be modeled by using a formal polynomial function of the two variables n and ε. Hence, the number of iterations can be modeled and estimated as a function $f(n, \varepsilon)$ of the two variables n and ε. In each improvement, the time complexity of finding a new center vector on the k clusters in ε-FKGC is $O(n^2)$ and the time complexity of finding a new ε-approximate fuzzy matrix in ε-FKGC is $O(n^2)$. Clearly, the time complexity of completing the algorithm ε-FKGC is $O(n^2 f(n, \varepsilon))$. Hence, the time complexity of generating a ε-approximate fuzzy matrix in FKGC is $O(n^4+n^2 f(n, \varepsilon))$.

For the vertex assignment in k-way (δ, γ)-balanced partitioning, first, based on the fuzzy memberships in the ε-approximate fuzzy matrix M, the maximum capacity δ, and the maximum size tolerance γ, the time complexity of constructing k partitions in the initial assignment is $O(n)$. Based on the assignment result of the k partitions in the initial assignment, the maximum capacity δ, and the maximum size tolerance γ, the time complexity of reassigning some vertices inside the k partitions in the iterative size modification is $O(n)$. Hence, the time complexity of completing the vertex assignment in k-way (δ, γ)-balanced partitioning is $O(n)$.

To sum up, the time complexity of completing the partitioning process in k-way (δ, γ)-balanced partitioning under capacity constraint δ and the size-tolerance constraint γ is $O(m+n^4+n^2 f(n, \varepsilon))$, where m is the number of gates and n is the number of qubits in a given QC.

IV. EXPERIMENTAL RESULTS

For k-way (δ, γ)-balanced partitioning in a DQC design, the proposed fuzzy-based partitioning algorithm has been implemented by using standard C++ language, compiled by gcc4.2.4 and run on an Intel Core i7-7700HQ CPU 3.80 GHz machine with 16 GB memory. In the experiments, eight tested circuits, Circuit01, Circuit02, Circuit03, Circuit04, Circuit05, Circuit06, Circuit07, and Circuit08, can be generated from the combination of some reversible circuits in the online resource, Revlib [31]. It is clear that the smaller the tolerant error ε in FKGC is, the higher the identification degree of the fuzzy memberships inside the final ε-approximate fuzzy matrix is and the more the running time in KBP is. To have the reasonable running time in KBP, the tolerant error ε in FKGC can be set as 0.001.

Basically, the convergence time on the generation of a final ε-approximate fuzzy matrix in FKGC seriously depends on the used initial fuzzy matrix. It is known that a better initial fuzzy matrix can lead to a better ε-approximate fuzzy matrix in FKGC. Because one edge with the largest weight is randomly selected in the construction of an initial partitioning result, the ε-approximate fuzzy matrix may not be unique. Hence, a set of ε-approximate fuzzy matrices in FKGC can be obtained by using a set of initial partitioning results. In the article, ten initial partitioning results can be used to find ten ε-approximate fuzzy matrices in FKGC and the ε-approximate fuzzy matrix with the least running time can be treated as the final fuzzy matrix in FKGC for any tested circuit.

To compare the communication cost in k-way (δ, γ)-balanced partitioning for a DQC design, Daei's recursive KL-based algorithm [21], and a modified partitioning algorithm, the combination of the genetic algorithm and the

IEEE Transactions on Quantum Engineering

TABLE 2. Experimental Results on CPU Time for the Fuzzy-Based Partitioning Algorithm Using Random Initial Partitioning or Our Proposed Initial Partitioning in KBP

Circuits	#qubits	#gates	k	δ	The Fuzzy-Based Partitioning Algorithm using Random Initial Partitioning				The Fuzzy-Based Partitioning Algorithm using Our Proposed Initial Partitioning			
					$\gamma = 1$		$\gamma = 2$		$\gamma = 1$		$\gamma = 2$	
					#CC	CPU Time (sec)	#CC	CPU Time (sec)	#CC	CPU Time (sec)	#CC	CPU Time (sec)
Circuit01	21	62	3	8	25	0.75(100%)	24	0.79(100%)	25	0.62(82.7%)	24	0.65(82.3%)
				9	25	0.81(100%)	24	0.84(100%)	25	0.64(79.0%)	24	0.68(81.0%)
			4	6	20	0.86(100%)	18	0.91(100%)	20	0.72(83.7%)	18	0.76(83.5%)
				7	20	0.92(100%)	18	0.97(100%)	20	0.78(84.8%)	18	0.85(87.6%)
Circuit02	27	74	3	10	29	0.97(100%)	27	1.03(100%)	29	0.80(82.5%)	27	0.84(81.6%)
				11	29	1.08(100%)	27	1.15(100%)	29	0.87(80.6%)	27	0.94(81.7%)
			4	7	23	0.99(100%)	23	1.07(100%)	23	0.91(92.0%)	23	0.96(89.7%)
				8	23	1.15(100%)	21	1.22(100%)	23	0.98(85.2%)	21	1.06(86.9%)
Circuit03	32	92	3	11	35	1.07(100%)	35	1.15(100%)	35	0.89(83.2%)	35	0.95(82.6%)
				12	35	1.20(100%)	35	1.34(100%)	35	0.93(77.5%)	35	0.99(73.9%)
			4	9	27	1.21(100%)	24	1.35(100%)	27	1.07(88.4%)	24	1.14(84.4%)
				10	27	1.32(100%)	24	1.42(100%)	27	1.12(84.8%)	24	1.20(84.5%)
Circuit04	34	106	3	12	40	1.19(100%)	37	1.32(100%)	40	1.06(89.1%)	37	1.13(85.6%)
				13	40	1.31(100%)	37	1.43(100%)	40	1.13(86.3%)	37	1.20(83.9%)
			4	9	32	1.32(100%)	30	1.47(100%)	32	1.14(86.4%)	30	1.20(81.6%)
				10	32	1.41(100%)	29	1.52(100%)	32	1.21(85.8%)	29	1.29(84.9%)
Circuit05	39	102	4	10	29	1.40(100%)	29	1.54(100%)	29	1.23(87.9%)	29	1.30(84.4%)
				11	29	1.52(100%)	29	1.62(100%)	29	1.29(84.9%)	29	1.35(83.3%)
			5	8	24	1.53(100%)	24	1.63(100%)	24	1.32(86.3%)	24	1.39(85.3%)
				9	24	1.64(100%)	21	1.73(100%)	24	1.39(84.8%)	21	1.47(85.0%)
Circuit06	43	136	4	11	39	1.59(100%)	39	1.71(100%)	39	1.31(82.4%)	39	1.39(81.3%)
				12	39	1.68(100%)	36	1.78(100%)	39	1.38(82.1%)	36	1.44(80.9%)
			5	9	32	1.67(100%)	29	1.86(100%)	32	1.40(83.8%)	29	1.48(79.6%)
				10	32	1.82(100%)	27	1.94(100%)	32	1.49(81.9%)	27	1.58(81.4%)
Circuit07	52	157	4	14	43	1.76(100%)	40	1.87(100%)	43	1.49(84.7%)	40	1.53(81.8%)
				15	43	1.89(100%)	40	1.95(100%)	43	1.52(80.4%)	40	1.58(81.0%)
			5	11	36	1.97(100%)	32	2.10(100%)	36	1.65(83.8%)	32	1.73(82.4%)
				12	36	2.01(100%)	29	2.14(100%)	36	1.75(87.1%)	29	1.84(86.0%)
Circuit08	63	197	4	16	53	2.04(100%)	53	2.15(100%)	53	1.69(82.8%)	53	1.74(80.9%)
				17	53	2.15(100%)	49	2.27(100%)	53	1.75(81.4%)	49	1.81(79.7%)
			5	13	43	2.19(100%)	38	2.30(100%)	43	1.77(80.8%)	38	1.84(80.0%)
				14	43	2.32(100%)	38	2.49(100%)	43	1.88(81.0%)	38	1.97(79.1%)

modified tuba-search algorithm from Dadkhah's partitioning algorithm [22] with the capacity constraint inside a partition as $\lfloor n/k \rfloor + 1$ or $\lfloor n/k \rfloor + 2$ and the size-tolerance constraint between two partitions as 3 can also be implemented for the eight tested circuits. Because of the bipartitioning feature in the KL-based algorithm, the value k ($k = 2^p$) must be restricted in Daei's recursive KL-based algorithm in k-way (δ, γ)-balanced partitioning, where p is the number of recursions. In Tables 2 and 3, "*#qubits*" denotes the number of qubits in a tested circuit, "*#gates*" denotes the number of gates in a tested circuit, "k" denotes the partitioning number in a tested circuit, "δ" denotes the maximum capacity inside each partition in KBP, "γ" denotes the maximum size tolerance between two partitions in KBP, "*#CC*" denotes the communication cost in KBP for a tested circuit, and "*CPU Time*" denotes the execution time for a tested circuit.

It is known that a good initial partitioning result can shorten the convergence time of a fuzzy matrix in FKGC. To measure the performance of our proposed initial partitioning result in k-way (δ, γ)-balanced partitioning, the fuzzy-based partitioning algorithm using a random initial partitioning result and the fuzzy-based partitioning algorithm using our proposed initial partitioning result in k-way (δ, γ)-balanced partitioning can be implemented and compared in the first experiment. For the eight tested circuits in the first experiment, the experimental results of the fuzzy-based partitioning algorithm using a random initial partitioning result and the fuzzy-based partitioning algorithm using our proposed initial partitioning result in k-way (δ, γ)-balanced partitioning for a DQC design can be obtained and listed in Table 2. It is assumed that two different capacity constraints $\delta = \lfloor n/k \rfloor + 1$ and $\delta = \lfloor n/k \rfloor + 2$ can be set in the experiment. Compared with the fuzzy-based partitioning algorithm using a random initial partitioning result with two different size-tolerance constraints $\gamma = 1$ and $\gamma = 2$ in 3-way, 4-way, or 5-way balanced partitioning, the experimental results show that the fuzzy-based partitioning algorithm using our proposed initial partitioning result with two different size-tolerance constraints $\gamma = 1$ and $\gamma = 2$ can reduce 16.0% and 17.3% of the CPU time to obtain the same communication cost for the eight tested circuits on the average, respectively.

To measure the communication cost of the fuzzy-based partitioning algorithm in k-way (δ, γ)-balanced partitioning, Daei's recursive KL-based algorithm [21], the modified partitioning algorithm from Dadkhah's partitioning algorithm [22], and the proposed fuzzy-based partitioning algorithm in k-way (δ, γ)-balanced partitioning can be implemented and compared in the second experiment. For the eight tested

TABLE 3. Experimental Results on Communication Cost for Daei's Recursive KL-Based Algorithm [21], Modified Partitioning Algorithm From Dadkhah's Partitioning Algorithm [22], and the Proposed Fuzzy-Based Partitioning Algorithm in KBP

Circuits	#qubits	#gates	k	δ	Daei's Recursive KL-based Algorithm [21]		Modified Partitioning Algorithm from [22]		The Proposed Fuzzy-Based Partitioning Algorithm					
									$\gamma = 1$		$\gamma = 2$		$\gamma = 3$	
					#CC	CPU Time (sec)	#CC	CPU Time (sec)	#CC	CPU Time (sec)	#CC	CPU Time (sec)	#CC	CPU Time (sec)
Circuit01	21	62	3	8	-	-	24(100%)	1.86	25(104.2%)	0.62	24(100%)	0.65	23(95.8%)	0.74
				9	-	-	24(100%)	1.72	25(104.2%)	0.64	24(100%)	0.68	22(91.7%)	0.78
			4	6	24(100%)	1.07	19(79.2%)	2.14	20(83.3%)	0.72	18(75.0%)	0.76	18(75.0%)	0.81
				7	24(100%)	1.07	18(75.0%)	2.08	20(83.3%)	0.78	18(75.0%)	0.85	16(66.7%)	0.91
Circuit02	27	74	3	10	-	-	27(100%)	2.52	29(107.4%)	0.80	27(100%)	0.84	25(92.6%)	0.87
				11	-	-	26(100%)	2.41	29(111.5%)	0.87	27(103.8%)	0.94	24(92.3%)	0.96
			4	7	29(100%)	1.39	24(82.8%)	2.69	23(79.3%)	0.91	23(79.3%)	0.96	23(79.3%)	1.01
				8	29(100%)	1.39	23(79.3%)	2.54	23(79.3%)	0.98	21(72.4%)	1.06	20(69.0%)	1.14
Circuit03	32	92	3	11	-	-	35(100%)	3.11	35(100%)	0.89	35(100%)	0.95	31(88.6%)	1.11
				12	-	-	34(100%)	3.02	35(102.9%)	0.93	35(102.9%)	0.99	30(88.2%)	1.17
			4	9	32(100%)	1.74	28(87.5%)	3.32	27(84.4%)	1.07	24(75.0%)	1.14	22(68.8%)	1.22
				10	32(100%)	1.74	27(84.4%)	3.19	27(84.4%)	1.12	24(75.0%)	1.20	21(65.6%)	1.26
Circuit04	34	106	3	12	-	-	39(100%)	3.26	40(102.6%)	1.06	37(94.9%)	1.13	35(89.7%)	1.13
				13	-	-	38(100%)	3.18	40(105.3%)	1.13	37(97.4%)	1.20	34(89.5%)	1.20
			4	9	36(100%)	1.89	33(91.7%)	3.53	32(88.9%)	1.14	30(83.3%)	1.20	30(83.3%)	1.27
				10	36(100%)	1.89	31(86.1%)	3.41	32(88.9%)	1.21	29(80.6%)	1.29	28(77.8%)	1.35
Circuit05	39	102	4	10	36(100%)	2.28	30(83.3%)	4.07	29(80.6%)	1.23	29(80.6%)	1.30	29(80.6%)	1.36
				11	36(100%)	2.28	29(80.6%)	3.92	29(80.6%)	1.29	29(80.6%)	1.35	25(69.4%)	1.43
			5	8	-	-	24(100%)	4.52	24(100%)	1.32	24(100%)	1.39	21(87.5%)	1.50
				9	-	-	24(100%)	4.21	24(100%)	1.39	21(87.5%)	1.47	20(83.3%)	1.62
Circuit06	43	136	4	11	46(100%)	2.43	40(87.0%)	4.21	39(84.8%)	1.31	39(84.8%)	1.39	39(84.8%)	1.45
				12	46(100%)	2.43	39(84.8%)	4.05	39(84.8%)	1.38	36(78.3%)	1.44	34(73.9%)	1.52
			5	9	-	-	32(100%)	4.67	32(100%)	1.40	29(90.6%)	1.48	27(84.4%)	1.54
				10	-	-	31(100%)	4.40	32(103.2%)	1.49	27(87.1%)	1.58	26(83.9%)	1.66
Circuit07	52	157	4	14	51(100%)	2.82	43(84.3%)	5.02	43(84.3%)	1.49	40(78.4%)	1.53	39(76.5%)	1.60
				15	51(100%)	2.82	42(82.3%)	4.89	43(84.3%)	1.52	40(78.4%)	1.58	38(74.5%)	1.64
			5	11	-	-	35(100%)	5.63	36(102.9%)	1.65	32(91.4%)	1.73	30(85.7%)	1.74
				12	-	-	34(100%)	5.41	36(105.9%)	1.75	29(85.3%)	1.84	27(79.4%)	1.81
Circuit08	63	197	4	16	62(100%)	3.13	53(85.5%)	6.26	53(85.5%)	1.69	53(85.5%)	1.74	53(85.5%)	1.81
				17	62(100%)	3.13	51(82.3%)	6.03	53(85.5%)	1.75	49(79.0%)	1.81	47(75.8%)	1.89
			5	13	-	-	42(100%)	7.05	43(102.4%)	1.77	38(90.5%)	1.84	36(85.7%)	2.01
				14	-	-	41(100%)	6.78	43(104.9%)	1.88	38(92.7%)	1.97	34(82.9%)	2.18

circuits in the second experiment, the experimental results of Daei's recursive KL-based algorithm [21], the modified partitioning algorithm from Dadkhah's partitioning algorithm [22], and the proposed fuzzy-based partitioning algorithm in k-way (δ, γ)-balanced partitioning can be obtained and listed in Table 3. It is assumed that two different capacity constraints $\delta = \lfloor n/k \rfloor + 1$ and $\delta = \lfloor n/k \rfloor + 2$ can be set in the experiment. Compared with Daei's recursive KL-based algorithm [21] in 4-way balanced partitioning, the experimental results show that the proposed fuzzy-based partitioning algorithm with three different size-tolerance constraints $\gamma = 1$, $\gamma = 2$, and $\gamma = 3$ can use 58.3%, 61.3%, and 64.5% of CPU time to reduce 16.1%, 21.2%, and 24.6% of the communication cost for the eight tested circuits on the average, respectively. Compared with the modified partitioning algorithm from Dadkhah's partitioning algorithm [22] in 3-way, 4-way, or 5-way balanced partitioning, the experimental results show that the proposed fuzzy-based partitioning algorithm with the size-tolerance constraint $\gamma = 3$ can use 35.0% of CPU time to reduce 11.1% of the communication cost for the eight tested circuits on the average, respectively.

V. CONCLUSION

Given a large QC in a DQC design, first, an edge-weighted connection graph can be constructed from the gates in the given QC. Furthermore, based on the edge connections in the connection graph, a given partitioning number k and a tolerant error ε , the probabilistic connection strength between two vertices in the connection graph can be estimated and the initial KGP result via bottom-up clustering can be obtained. Based on the definition of the clustering distance between two vertices in the connection graph, the ε-approximate fuzzy matrix in FKGC can be obtained. Finally, given the maximum capacity δ inside each partition and the maximum size tolerance γ between two partitions, all the vertices in the connection graph can be assigned onto k partitions to minimize the communication cost in k-way (δ, γ)-balanced partitioning for a DQC design.

In future works, the communication cost between two qubits must be discussed and analyzed according to the characterization of the utilized gates in a DQC design. In addition, the noise effect during the execution of quantum gates can be further considered in a DQC design.

REFERENCES

[1] C. Easttom, *Quantum Computing Fundamentals*. Reading, MA, USA: Addison-Wesley, 2021. [Online]. Available: https://www.pearson.com/store/p/quantum-computing-fundamentals/P200000007382/9780136793816

[2] N. Abdessaied and R. Drechsler, *Reversible and Quantum Circuits: Optimization and Complexity Analysis*. Berlin, Germany: Springer, 2016, doi: 10.1007/978-3-319-31937-7.

[3] T. S. Humble, H. Thapliyal, E. Munoz-Coreas, F. A. Mohiyaddin, and R. S. Bennink, "Quantum computing circuits and devices," *IEEE Des. Test*, vol. 36, no. 3, pp. 69–94, Jun. 2019, doi: 10.1109/MDAT.2019.2907130.

[4] R. Van Meter and S. J. Devitt, "The path to scalable distributed quantum computing," *Computer*, vol. 49, no. 9, pp. 31–42, Sep. 2016, doi: 10.1109/MC.2016.291.

[5] R. Cleve and H. Buhrman, "Substituting quantum entanglement for communication," *Phys. Rev. A*, vol. 56, pp. 1201–1204, 1997, doi: 10.1103/PhysRevA.56.1201.

[6] J. I. Cirac, A. K. Ekert, S. F. Huelga, and C. Macchiavello, "Distributed quantum computation over noisy channels," *Phys. Rev. A*, vol. 59, pp. 4249–4254, 1999, doi: 10.1103/PhysRevA.59.4249.

[7] R. Beals et al., "Efficient distributed quantum computing," *Proc. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 469, 2013, Art. no. 20120686, doi: 10.1098/rspa.2012.0686.

[8] J. Yepez, "Type-II quantum computers," *Int. J. Modern Phys. C*, vol. 12, no. 9, pp. 1273–1284, 2001, doi: 10.1142/S0129183101002668.

[9] H. K. Lo, "Classical-communication cost in distributed quantum-information processing: A generalization of quantum-communication complexity," *Phys. Rev. A*, vol. 62, 1999, Art. no. 012313, doi: 10.1103/PhysRevA.62.012313.

[10] M. Caleffi, A. S. Cacciapuoti, and G. Bianchi, "Quantum Internet: From communication to distributed computing," in *Proc. 5th ACM Int. Conf. Nanoscale Comput. Commun.*, 2018, pp. 1–4, doi: 10.1145/3233188.3233224.

[11] A. S. Cacciapuoti, M. Caleffi, F. Tafuri, F. S. Cataliotti, S. Gherardini, and G. Bianchi, "Quantum Internet: Networking challenges in distributed quantum computing," *IEEE Netw.*, vol. 34, no. 1, pp. 137–143, Jan./Feb. 2020, doi: 10.1109/MNET.001.1900092.

[12] A. Yimsiriwattana and S. J. Lomonaco, "Distributed quantum computing: A distributed Shor algorithm," *Proc. SPIE*, vol. 5436, pp. 360–372, 2004, doi: 10.1117/12.546504.

[13] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Rev.*, vol. 41, no. 2, pp. 303–332, 1999, doi: 10.1137/S0036144598347011.

[14] R. Van Meter, W. J. Munro, K. Nemoto, and K. M. Itoh, "Arithmetic on a distributed-memory quantum multicomputer," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 3, 2008, Art. no. 2, doi: 10.1145/1324177.1324179.

[15] M. Zomorodi-Moghadam, M. Houshmand, and Monireh Houshmand, "Optimizing teleportation cost in distributed quantum circuits," *Int. J. Theor. Phys.*, vol. 57, no. 3, pp. 848–861, 2018, doi: 10.1007/s10773-017-3618-x.

[16] M. Houshmand, Z. Mohammadi, M. Zomorodi-Moghadam, and M. Houshmand, "An evolutionary approach to optimizing teleportation cost in distributed quantum computation," *Int. J. Theor. Phys.*, vol. 59, pp. 1315–1329, 2020, doi: 10.1007/s10773-020-04409-0.

[17] D. Dadkhah, M. Zomorodi, and S. E. Hosseini, "A new approach for optimization of distributed quantum circuits," *Int. J. Theor. Phys.*, vol. 60, pp. 3271–3285, 2021, doi: 10.1007/s10773-021-04904-y.

[18] I. Ghodsollahee, Z. Davarzani, M. Zomorodi, P. Pławiak, M. Houshmand, and Mahboobeh Houshmand, "Connectivity matrix model of quantum circuits and its application to distributed quantum circuit optimization," *Quantum Inf. Process.*, vol. 20, 2021, Art. no. 235, doi: 10.1007/s11128-021-03170-5.

[19] P. Andrés-Martínez and C. Heunen, "Automated distribution of quantum circuits via hypergraph partitioning," *Phys. Rev. A*, vol. 100, 2019, Art. no. 032308, doi: 10.1103/PhysRevA.100.032308.

[20] Z. Davarzani, M. Zomorodi-Moghadam, M. Houshmand, and M. Nouri-Bayg, "A dynamic programming approach for distributing quantum circuits by bipartite graphs," *Quantum Inf. Process.*, vol. 19, 2020, Art. no. 360, doi: 10.1007/s11128-020-02871-7.

[21] O. Daei, K. Navi, and M. Zomorodi-Moghadam, "Optimized quantum circuit partitioning," *Int. J. Theor. Phys.*, vol. 59, pp. 3804–3820, 2020, doi: 10.1007/s10773-020-04633-8.

[22] D. Dadkhah, M. Zomorodi, S. E. Hosseini, P. Plawiak, and X. Zhou, "Reordering and partitioning of distributed quantum circuits," *IEEE Access*, vol. 10, pp. 70329–70341, 2022, doi: 10.1109/ACCESS.2022.3186485.

[23] J.-T. Yan and P.-Y. Hsiao, "A fuzzy clustering algorithm for graph bisection," *Inf. Process. Lett.*, vol. 52, no. 5, pp. 259–263, 1994, doi: 10.1016/0020-0190(94)00148-0.

[24] J.-T. Yan and P.-Y. Hsiao, "A new fuzzy-clustering-based approach for two-way circuit partitioning," in *Proc. 8th Int. Conf. VLSI Des.*, 1995, pp. 359–364, doi: 10.1109/ICVD.1995.512139.

[25] J.-T. Yan, "Connection-oriented net model and fuzzy clustering techniques for k-way circuit partitioning," in *Proc. IEEE Int. Conf. Comput. Des.*, 1995, pp. 236–241, doi: 10.1109/ICCD.1995.528816.

[26] J.-T. Yan, "Fuzzy-clustering-based algorithm for circuit partitioning in standard cell placement," *Electron. Lett.*, vol. 31, no. 3, pp. 151–152, 1995, doi: 10.1049/el:19950121.

[27] J.-T. Yan, "Fuzzy-clustering-based circular topological via minimization in PCB designs," *IEEE Trans. Fuzzy Syst.*, vol. 29, no. 5, pp. 1023–1036, May 2021, doi: 10.1109/TFUZZ.2020.2968857.

[28] J. C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," *J. Cybern.*, vol. 3, pp. 32–57, 1973, doi: 10.1080/01969727308546046.

[29] R. L. Cannon, J. V. Dave, and J. C. Bezdek, "Efficient implementation of the fuzzy *c*-means clustering algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 2, pp. 248–255, Mar. 1986, doi: 10.1109/TPAMI.1986.4767778.

[30] M. A. Ismail and S. Z. Selim, "Fuzzy *c*-means: Optimality of solutions and effective termination of the algorithm," *Pattern Recognit.*, vol. 19, pp. 481–485, 1986, doi: 10.1016/0031-3203(86)90048-8.

[31] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An online resource for reversible functions and reversible circuits," in *Proc. 38th Int. Symp. Mult. Valued Log.*, 2008, pp. 220–225, doi: 10.1109/ISMVL.2008.43.

Jin-Tai Yan (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer and information science from National Chiao-Tung University, Hsinchu, Taiwan, in 1988, 1990, and 1995, respectively.

From 1997 to 1999, he was a Postdoctoral Researcher with Advanced CPU Technology, Microelectronics and Information Systems Research, National Chiao-Tung University, Hsinchu, Taiwan. From 1999 to 2019, he was a faculty member with the Department of Computer Science and Information Engineering, Chung-Hua University, Hsinchu, Taiwan. From 2019 to 2022, he was a Professor Researcher with the Office of Research and Development, Tainan National University of the Arts, Tainan, Taiwan. He is currently a Professor with the Graduate Institute of Sound Technology, Tainan National University of the Arts, Tainan, Taiwan. His current research interests include electronic design automation, quantum circuit design, digital integrated circuit design, and computer architecture.