Distributing Quantum Circuits Using Teleportations

Ranjani G. Sundaram
Stony Brook University, NY
rasundaram@cs.stonybrook.edu

Himanshu Gupta
Stony Brook University, NY
hgupta@cs.stonybrook.edu

***Abstract*—Scalability is currently one of the most sought-after objectives in the field of quantum computing. Distributing a quantum circuit across a quantum network is one way to facilitate large computations using current quantum computers. In this paper, we consider the problem of distributing a quantum circuit across a network of heterogeneous quantum computers, while minimizing the number of teleportations (the communication cost) needed to implement gates spanning multiple computers. We design two algorithms for this problem. The first, called Local-Best, initially distributes the qubits across the network, then tries to teleport qubits only when necessary, with teleportations being influenced by gates in the near future. The second, called Zero-Stitching, divides the given circuit into sub-circuits such that each sub-circuit can be executed using zero teleportations and the teleportation cost incurred at the borders of the sub-circuits is minimal. We evaluate our algorithms over a wide range of randomly-generated circuits as well as known benchmarks, and compare their performance to prior work. We observe that our techniques outperform the prior approach by a significant margin (up to 50%).**

I. Introduction

Quantum computing is a new method of computation that has the potential to solve some problems deemed intractable using classical computation. Quantum computers (QCs) use the properties of quantum mechanics to analyze large numbers of possibilities and extract potential solutions to complex problems. However, the same properties raise challenges when we try to scale QCs to perform computations involving a large number of qubits. One approach to tackle these challenges is to distribute such computations across a quantum network (QN).

We seek to design methods that distribute a given quantum program over a quantum network efficiently. A natural optimization is to minimize the overall communication cost incurred in executing the distributed program over the given network. This problem is known to be NP-Hard, and has been considered recently in several works [3, 5, 12]. There are two main means of viable quantum communication, viz., teleportations and cat-entanglements. Cat-entanglements, in essence, create read-only copies of a qubit; these copies can thus be used as control operands to binary quantum gates. Though cat-entanglements facilitate higher distribution/concurrency, they have many disadvantages: (i) The copies need to be disentangled (i.e., destroyed) prior to any unary operation on the original qubit. (ii) Distribution algorithms based on cat-entanglements have assumed circuits composed of only binary CZ (due to their symmetry) and unary gates. (iii) The shared copies, being entangled, incur a higher degree of decoherence. Thus, it is desirable to restrict communication modes to only teleportations. In this work, we thus address the problem of distribution of quantum circuits using only teleportations.

This work was supported in part by the National Science Foundation under Award FET-2106447 and Award CNS-2128187.

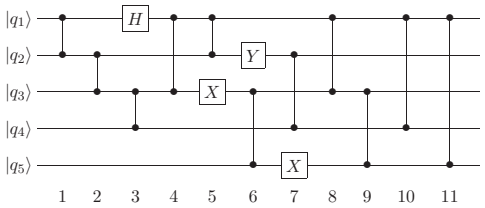

Fig. 1: Quantum Circuit Representation.

Our Contributions. In this work, we address the DQC−T problem, which aims to distribute a given quantum circuit across a QN with a minimum number of teleportations. DQC−T has been addressed before: [6, 18, 24] provide worst-case exponential-time algorithm under specialized settings, while [5] provides a polynomial-time heuristic for the special case when the quantum computers in the network have identical memory capacities. In this work, we design two novel polynomial-time algorithms for the general DQC−T problem that outperforms the algorithm in [5] by up to 50%.

II. Background

We start with giving a brief background on two key quantum concepts relevant to our paper: quantum circuits and quantum communication methods.

Quantum Circuits. Quantum computation is typically abstracted as a *circuit*, where horizontal "wires" represent *qubits* which carry quantum data, and operations on the qubits performed by vertical "gates" connecting the operand wires [17] (See Fig. 1). Quantum computers (QCs) evaluate a circuit by applying the gates in the left-to-right order, so this circuit can also be understood as a sequence of machine-level instructions (gates) over a fixed number of data cells (qubits).

Representation. A universal gate set is a set of gates sufficient to represent any quantum computation. We consider the universal gate set with unary and $CNOT$ gates. We represent an *abstract quantum circuit* C over a set of qubits $Q = \{q_1, q_2, \ldots\}$ as a sequence of gates $\langle g_1, g_2, \ldots\rangle$ where each g_t is either binary $CNOT$ gate or a unary gate. We represent binary gates in a circuit as triplets (q_i, q_j, t) where q_i and q_j are the two operands, and t is the time instant (see below) of the gate in the circuit, and unary gates as pairs (q_i, t) where q_i is the operand and t is the time instant. We use N_q and N_g to denote the number of qubits and gates in the circuit.

Time Instants. Each gate occurs uniquely at a time instant. In addition to the instants where the gates occur, we introduce time instants in between consecutive gates for convenience; we

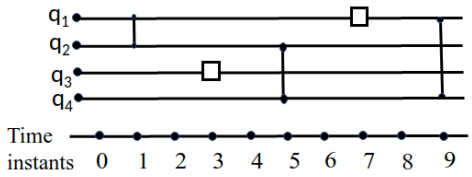

Fig. 2: The gates are at odd-numbered time instants $1, 3, \ldots, 9$. Even-numbered time instants are added to allow for potential teleportations.

enforce that teleportations occur only at these additional time-instants so that they don't interfere (or occur concurrently) with the gate operations. See Fig. 2.

Quantum Communication. If a given quantum circuit is to be evaluated in a distributed fashion over a network of QCs, we have to first distribute the qubits over the QCs. But such a distribution may induce gates in the circuit to span multiple QCs. To execute such *non-local* gates, we need to bring all operands' values into a single QC via quantum communication. However, direct/physical transmission of quantum data is subject to unrecoverable errors, as classical procedures such as amplified signals or re-transmission cannot be applied due to quantum no-cloning [9, 22].[1] Fortunately, there are other viable ways to communicate qubits across network nodes, as described below.

Teleportation. An alternative approach to physically transmit qubits is via *teleportation* [4] which requires an a priori distribution of maximally-entangled pair (MEP) of qubits (e.g., Bell Pair) over the two nodes [13, 14]. With an MEP distributed over nodes A and B, teleportation of a qubit state from A to B can be accomplished using classical communication and local gate operations, while consuming/destroying the MEP.

Cat-Entanglement: Creating "Linked Copies" of a Qubit. Another means of communicating qubit states is by creating *linked read-only copies* of a qubit across QCs, via *cat-entanglement* operations [11, 23] which, like teleportation, require a Bell Pair to be shared *a priori*. These linked copies are useful in efficient distributed evaluation of circuits involving only *CZ* and unary gates, as a single copy can be used as a control qubit for a contiguous sequence of *CZ* gates. These linked copies can only be used till there is a unary operation on the qubit.

However, restricting to circuits involving only *CZ* and unary gates means not exploiting potential circuit representations that contain fewer binary gates. Moreover, cat-entanglements entail the creation of entangled qubits that remain entangled for extended periods of time; these entangled qubits have a higher chance of decoherence and can lead to loss of fidelity of original qubits. Thus, in this work, we use only teleportations.

III. Problem Formulation and Related Work

We start by defining some terms and models before moving on to the problem formulation.

[1]Quantum error correction mechanisms [8, 16] can be used to mitigate the transmission errors, but their implementation is very challenging and is not expected to be used until later generations of quantum networks.

Quantum Network (QN). We represent a QN (e.g., a quantum sensor network [7, 15]) as a connected undirected graph with nodes representing QCs and edges representing (quantum and classical) direct communication links. Nodes of the network are denoted by P; we use the words node, computer, and QC interchangeably. We denote the number of computers/nodes in the network by N_p. Each node $p \in P$ has a storage capacity s_p that represents the maximum number of qubits that can exist in node p at any time instant.

Home-Computer (Function). To distribute the circuit, we first distribute the qubits of the circuit among the quantum computers. Later, our algorithms determine how to move these qubits around via teleportations, but, at any given time instant, each qubit is contained in a unique computer. Thus, at every time instant t, we can define a function $\pi^t : Q \rightarrow P$ from qubits to the computers in the quantum network. This function specifies which computer a qubit is contained in at a time instant and is called the *Home-Computer function*.

Naturally, at any time instant, the home computer function must satisfy the storage capacity constraints of every computer. Therefore, for a computer p and a time instant t, the number of qubits such that $\pi^t(q) = p$ must be less than s_p.

Non-Local Gates. A gate (q_i, q_j, t) is considered to be *non-local* at time t if q_i and q_j have different home-computers, i.e., $\pi^t(q_i) \neq \pi^t(q_j)$. Other gates are considered local gates.

Communication (Teleportation) Cost. We represent a teleportation by a quadruplet (q, p_1, p_2, t) which signifies that the qubit q is teleported from computer p_1 to computer p_2 at time t. Note that p_1 must be equal to $\pi^{t-1}(q)$, and the teleportation (q, p_1, p_2, t) results in $\pi^t(q)$ becoming p_2. As mentioned before, we restrict teleportations to only occur at non-gate time instants.

The cost of a teleportation (q, p_1, p_2, t) is defined as the distance between the nodes p_1 and p_2 in the quantum network graph, i.e., minimum number of hops between p_1 and p_2 in the network graph. This cost accounts for the fact that teleporting a qubit from p_1 to p_2 requires an EP over nodes p_2 and p_1 whose generation cost we assume to be proportional to the distance between p_1 and p_2. For simplicity, we assume that teleportation across two remote nodes does not use any storage space in the intermediate nodes.

`DQC-T` **Problem.** Given a quantum network and a quantum circuit, our objective is to find a valid home-computer function at each time instant of the circuit such that (i) all gate operations of the circuit are local, and (ii) the total teleportation cost incurred (due to changes in the home-computer function over time) is minimized.

Note that teleportations incurred can be uniquely determined from the home-computer function for every instant. The `DQC-T` problem can be shown to be NP-Hard by a reduction from the graph bisection problem. We omit the proof here.

Example 1. Consider a `DQC-T` problem instance in Fig. 3—a circuit with four qubits and two computers each with a storage memory of three qubits. We assume each pair of computers to be connected by a network link, and thus, the

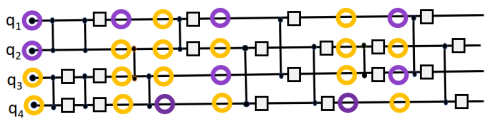

Fig. 3: Solution to a DQC−T instance.

cost of any teleportation is one. A solution for this instance is also illustrated in the figure. Initially, qubits q_1 and q_2 are assigned to the first computer (signified by purple circles), while q_3 and q_4 are assigned to the second computer (signified by yellow circles). As non-local gates are encountered, the home-computer function changes. Each time a particular qubit changes color, one teleportation occurs. The total cost of the solution provided is 5 (the number of teleportations).

Related Work. Daei et al. [5] consider the problem of reducing teleportation cost in distributed quantum circuits. The main idea behind their paper is to construct an appropriate graph over qubits and use the Kernighan-Lin graph bisection algorithm iteratively to partition the graph into k parts. Then, each part is assigned to a different quantum computer, and every non-local gate is executed using teleportations. In another work, Moghadam et al. [24] consider the special case of only two computers with the qubits assignment across the computers is already given; for this special case, they provide an exponential time algorithm that finds the minimum number of teleportations to execute all binary gates. Finally, Nikahd et al. [18] first segregate the binary gates into "levels," then determine the best partition of qubits for each level by solving an integer linear program.

Some works have addressed the DQC problem using cat-entanglements as the only means of communication. For example, [3] poses the DQC problem as a balanced hypergraph partitioning problem and gives a heuristic based on hypergraph min-cut. For a simplified setting, [12] presents a two-step algorithm for the DQC problem, wherein the first step determines the partitioning of qubits to computers through balanced graph partitioning and the second step minimizes the number of cat-entanglement operations via an iterative greedy approach. Finally, in [20], the authors allow use of *both* teleportations and cat-entanglements to communicate quantum states, and consider the general DQC problem with both storage and execution memory constraints on computers. In this work, we focus on using only teleportations due to reasons mentioned in Section II.

IV. LOCAL-BEST Algorithm

We start by describing our first algorithm, LOCAL-BEST.

Basic Idea. The high-level idea behind the LOCAL-BEST algorithm is to teleport qubits only when necessary, with teleportations being influenced by gates in the near-future. LOCAL-BEST algorithm has two steps: (i) Finding an *initial* assignment of qubits to computers such that the number of resulting non-local binary gates is minimized, and (ii) For each non-local binary gate g, choose the teleportations to execute g locally; the teleportations are chosen based on the "near future" such that the total number of teleportations is minimal.

Step 1. Initial Assignment of Qubits to Computers. Similar to the approach in [20], we assign qubits to computers using a Tabu Search [19] heuristic, in a way that satisfies the storage constraints at each node and minimizes the number of non-local gates. Below, we define the three key aspects that characterize the Tabu Search: (i) a solution, (ii) a solution's neighbors, and (iii) the cost of a solution; here, we use a simpler cost model than that used in [20].

Solution and Its Neighbors. In our context, a solution is any valid home-computer function, which satisfies each node's storage constraint. Neighbors of a given solution π can be defined as valid solutions π' that result from either: (i) changing the assignment/mapping of a single qubit without violating the storage constraints, or (ii) "swapping" of two qubits mapped to two different computers in π.

Solution's Cost. A solution's cost can be defined as the number of non-local gates resulting from the solution's qubit assignment. More formally, the cost of a solution π, denoted by $cost(\pi)$ is

$$cost(\pi) = \sum_{q_1, q_2 \in Q} w(q_1, q_2) \times \text{distance}(\pi(q_1), \pi(q_2))$$

where $w(q_1, q_2)$ is the *number* of binary gates between q_1 and q_2 if they are assigned to different computers.

Based on the above concepts, we can do the initial assignment of qubits using Tabu Search as follows.

Tabu Algorithm.

1) $\pi^* = \pi =$ initial random solution
2) $L = [\,]$ /* a bounded-length list of forbidden solutions */
3) Repeat for λ iterations:
   a) $\pi = argmin_{\pi' \in neighbors(\pi) - L}\ cost(\pi')$
   b) $\pi^* = \pi$ if $cost(\pi) < cost(\pi^*)$
   c) $L = L \cup \{\pi\}$, removing the oldest element from L if necessary to maintain length bound.
4) Return π^*

Step 2. Determining Teleportations. We now describe a procedure that determines a small number of teleportations to execute the non-local binary gates. The high-level idea of our approach is as follows:

- Scan the circuit from left to right and determine teleportations to execute each non-local gate encountered. Here, a gate is considered non-local or local based on the home-function at *that* time instant.
- For a non-local binary gate g encountered, determine the "best" computer where g should be executed by teleporting its operands to. Here, the best computer is determined (see below) based on the next (following) r binary gates that share an operand with g.

Best Computer to Execute g. More formally, to determine the best computer for a non-local gate g, we define and compute a "benefit" for each computer and pick the computer with the highest benefit. For a non-local binary gate $g = (q_1, q_2)$, the benefit of a computer p is define as the number of gates among the next r binary gates involving q_1 and/or q_2 that

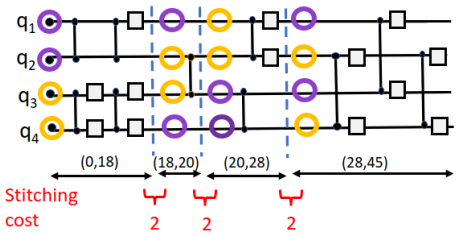

Fig. 4: Division of a circuit into four zero-cost sub-circuits $(0, 18), (18, 20), (20, 28)$, and $(28, 45)$, and stitching costs for the adjacent pairs of zero-cost sub-circuits.

become local (from non-local) by teleporting q_1 and q_2 to p. We note that in order to execute (q_1, q_2) in computer p, both q_1 and q_2 must be in p, and teleporting q_1 and/or q_2 would require one or two free storage spaces in p. If needed, the desired storage space(s) in p can be created by teleporting some other qubits from p to another computer. For simplicity, such supplementary teleportations are not taken into account in computing p's benefit. See [21] for the pseudo-code of the algorithm.

V. ZERO-STITCHING Algorithm

We now describe the ZERO-STITCHING algorithm which consists of two high-level steps: (i) identifying "zero-cost" sub-circuits, i.e., contiguous sub-circuits that can be executed without any teleportations; (ii) dividing the given circuit into zero-cost sub-circuits and "stitching" them together using teleportations. We start with defining a few concepts that will help in formalizing the overall ZERO-STITCHING algorithm.

Sub-Circuit. A sub-circuit of a given circuit C is any contiguous part of C, and can be represented by the starting and the ending time-instant. For example, in Fig. 4, $(0, 18), (18, 20), (20, 28)$, and $(28, 45)$ are some of the sub-circuits.

Zero-cost Sub-circuit (ZCSC). We deem a sub-circuit C to be *zero-cost* if, for *some* fixed home-computer function π, all gates in C become local; here, by *fixed* we mean that the home-computer function π remains unchanged throughout the sub-circuit C. We use the acronym ZCSC for zero-cost sub-circuit. In Fig. 4, the given circuit has been divided into four ZCSCs: $(0, 18), (18, 20), (20, 28)$, and $(28, 45)$, with two network nodes. Here, for each of the four ZCSCs, the figure also gives the home-computer function (represented by yellow and purple circles) which makes all the gates local.

Stitching Two ZCSCs. Consider two (adjacent) ZCSCs C_1 and C_2, and the corresponding home-computer functions π_1 and π_2 that make all their gates local. Then, stitching C_1 to C_2, for the given home-computer functions π_1 and π_2, signifies determining and using a set of teleportations that transform π_1 to π_2; the stitching cost is considered to be equal to the teleportation cost.

For example, in Fig. 4, stitching the ZCSC $(0, 18)$ to $(18, 20)$ incurs two teleportations: teleporting q_2 to the yellow computer and q_4 to the purple computer. Note that stitching of minimum cost for a given pair of ZCSCs and corresponding home-computer functions is trivial to determine. However, if we allow "relabelling" of computers, determining the minimum-cost stitching requires computing the maximum-weight matching in an appropriate bipartite graph (see the SCB() function below).

Using the above concepts, we now describe the two steps of ZERO-STITCHING algorithm.

Step 1. Identifying ZCSCs. In this step, we consider all possible sub-circuits of C and, for each sub-circuit C_i, we try to determine a home-computer function that makes all gates of C_i local. Since determining such a home-computer function is NP-Hard, we use a heuristic based on bin-packing approximation algorithm as described below.

For every sub-circuit C_i of given circuit C, we do the following:

- Construct a graph G_{C_i} with the given circuit's qubits as vertices, and an edge between two qubits if there are gates between the two qubits in the sub-circuit C_i.
- Consider the connected components of G_{C_i}.
- "Pack" the connected components into the given quantum computers, such that the qubits of any connected component are all within a single computer and the total number of qubits in any computer p is less than p's storage capacity. If we can find such a packing, then C_i is considered a *zero-cost sub-circuit*. Note that determining such a packing is equivalent to solving the well-studied *bin-packing problem* with computers as bins of size equal to their storage capacities and connected components of G_{C_i} as items of size equal to the number of qubits in the connected component. In our work, we use the first-fit greedy algorithm [10] to determine the packing of connected components into computers.

The above procedure determines a set of ZCSCs. For each ZCSC C_i, the packing algorithm also gives the (fixed) home-computer function π_i which makes all gates local; we refer to π_i as the *pre-computed* home-function for C_i.

Step 2. Efficient Division of C into ZCSCs. We use dynamic programming to find a division of C into a sequence of ZCSCs and their corresponding home-computer functions such that the total cost of stitching adjacent sub-circuits is minimal. We first present a simpler version of the dynamic programming (DP) procedure, and then improve it further.

SIMPLE DP: We can divide a given circuit C into ZCSCs with minimum aggregate stitching cost using dynamic programming as follows. Let $S[1, i, j]$ be the optimal cost for the sub-circuit $(1, j)$, formed by stitching together multiple ZCSCs within $(1, j)$ with the constraint that (i, j) is the last such ZCSC. Our goal is to determine $\min_i S[1, i, m]$, where m is the last time instant. We determine $S[\,]$ values recursively using dynamic programming as follows.

$$S[1, i, j] = min_{i'}\ S[1, i', i] + \text{SC}((i', i), (i, j)) \qquad (1)$$

Above, $\text{SC}((i', i), (i, j))$ is a function that gives the cost of stitching the ZCSC (i', i) to (i, j) for the given pre-computed home-computer functions. As mentioned above, the function

SC() is straightforward if we do not allow relabelling of computers. To allow relabelling of computers, we define a more general stitching function called SCB() that uses a maximum matching algorithm. Note that relabelling of computers is possible only when the computers have uniform storage capacity. We describe SCB() below.

SCB(C_1, C_2).

1) Let π_{C_1} and π_{C_2} be the pre-computed home-computer functions associated with C_1 and C_2 respectively.
2) Denote by P and $\bar{P}$ the partition of qubits corresponding to π_1 and π_2 respectively. Here, each element of P or $\bar{P}$ is a set of qubits mapped to a particular network node.
3) Construct a bipartite graph over elements of P and $\bar{P}$ as vertices. For each pair of elements P_i and $\bar{P}_j$ from P and $\bar{P}$ respectively, we associate an edge weight of $w(P_i, \bar{P}_j)$ equal to the number of qubits in both P_i and $\bar{P}_j$.
4) Find a maximum-weight matching M in the above graph.
5) Determine teleportations corresponding to the edges in the matching M.

We now motivate and then discuss an improved version of the above SIMPLE-DP algorithm that recomputes the home-computer function for sub-circuit (i, j) in Eqn. 1.

Motivation for Further Improvement. The above SIMPLE-DP algorithm yields a solution to the DQC−T problem by computing $min_{i'} S[1, i', m]$, where m is the number of binary gates. In fact, it is an optimal way to divide the input circuit into ZCSCs *given the pre-computed* home-functions for the ZCSCs. However, it is possible that a different set of home-functions for the ZCSCs may yield a better (i.e., with lower total stitching cost) DQC−T solution. Note that the pre-computed home-functions are computed independently for each ZCSC, while, for an optimal DQC−T solution, the home-computer functions of the ZCSCs being stitched should be as "close" to each other as possible. To incorporate the above insight, we modify the above SIMPLE-DP algorithm as below.

IMPROVED DP. Consider Eqn. 1. In the IMPROVED-DP algorithm, we allow re-computation of the home-computer function corresponding to the ZCSC (i, j) based on the home-computer function of the preceding ZCSC (i', i), when computing the optimal cost $S[1, i, j]$ for $(1, j)$. This modification is incorporated into the improved stitching function SC*, as described momentarily.

As before, let $S[1, i, j]$ be the optimal cost for the sub-circuit $(1, j)$, formed by stitching together multiple ZCSCs within $(1, j)$ with the constraint that (i, j) is the last such ZCSC. Our goal is to determine $\min_i S[1, i, m]$, where m is the last time instant. The improved dynamic programming formulation to determine $S[\,]$ values recursively is as follows.

$$S[1, i, j] = min_{i'}\ S[1, i', i] + \text{SC}^*(\pi_{(i',i)}, (i, j)). \quad (2)$$

Above, $\pi_{(i',i)}$ is the home-computer function of the sub-circuit (i', i) in the solution $S[1, i', i]$. The improved stitching cost function SC* works in two steps: (i) For the input operand (i, j) sub-circuit, it computes a home-computer function that "matches" as closely as possible with the input $\pi_{(i',i)}$—so that the cost of stitching the sub-circuit (i', i) to (i, j) is minimized, and (ii) it determines the teleportations required to stitch the sub-circuit (i', i) to (i, j) using the above home-computer functions.

The final improved stitching function SC*() is as follows.[2]

SC*(π_{C_1}, C_2):

1) Let π_{C_1} be the home-computer function associated with C_1 (given as input).
2) Compute π_2 for C_2 (as described below).
3) Compute teleportations that transform π_{C_1} to π_{C_2}.

If we do not allow relabelling of computers, then the last step is straightforward. If we do allow relabelling (which is only possible for balanced nodes), we replace the last step of SC* with Steps 2-5 of SCB; we call the resulting algorithm SCB*.

Computing π_2 for C_2. We compute π_2 for C_2 in the second step of SC* as follows. Essentially, we seek to compute a home-computer function π_2 for C_2 based on the home-computer function π_1 such that (i) π_2 makes all the gates of C_2 local, and (ii) the cost of stitching π_1 to π_2 is minimized. Consider an undirected graph G_{c2} over qubits in C_2 wherein there is an edge (q_i, q_j) if and only if there is a binary gate between q_i and q_j. To satisfy the first condition above (i.e., to make all gates of C_2 local), all the qubits in each connected component of G_{c2} must entirely lie in one computer. Now, to minimize the cost of stitching π_1 to π_2, we consider a weighted graph G'_{c2} over the connected components of G_{c2} as vertices, wherein the weight between two connected components cc_1 and cc_2 signify the cost of keeping them in two different network nodes/computers.[3] Then, we find an assignment of connected components of C_2 (i.e., vertices of G'_{c2}) to the network nodes, by partitioning the vertices into k sets (where k is the number of network nodes) such that the cut over these partitions is minimized and the number of qubits mapped to i^{th} set is less than the storage capacity of i^{th} computer. We find such an assignment using a generalized version of the Tabu search algorithm described in Section IV.

VI. Evaluation

In this section, we evaluate our algorithms over both randomly generated and some benchmark quantum circuits.

Algorithms Compared. We primarily compare three algorithms: (i) LOCAL-BEST from §IV, (ii) ZERO-STITCHING from §V (which uses SCB*), and (iii) REPEATED BISECTION from [5] which uses the Kernighan-Lin graph-bisection algorithm iteratively and implements non-local binary gates using two teleportations each. Other prior works on the DQC−T

[2]In function SC, the home-computer functions for the operands were implicit (i.e., the pre-computed home-computer functions), while in SC* we need to explicitly pass the home-computer function of (i', i), based on which the home-computer function of (i, j) is recomputed. This is facilitated by storing two values associated with a solution $S[1, i, j]$—the cost of $(1, j)$ and the home-computer function for (i, j); the latter is computed within the SC* function for the best i' in Eqn. 2.

[3]The weight between two connected components cc_1 and cc_2 is set to be the number of pairs of qubits (q_1, q_2) such that q_1 is in cc_1, q_2 is in cc_2 and $\pi(q_1) = \pi_1(q_2)$.

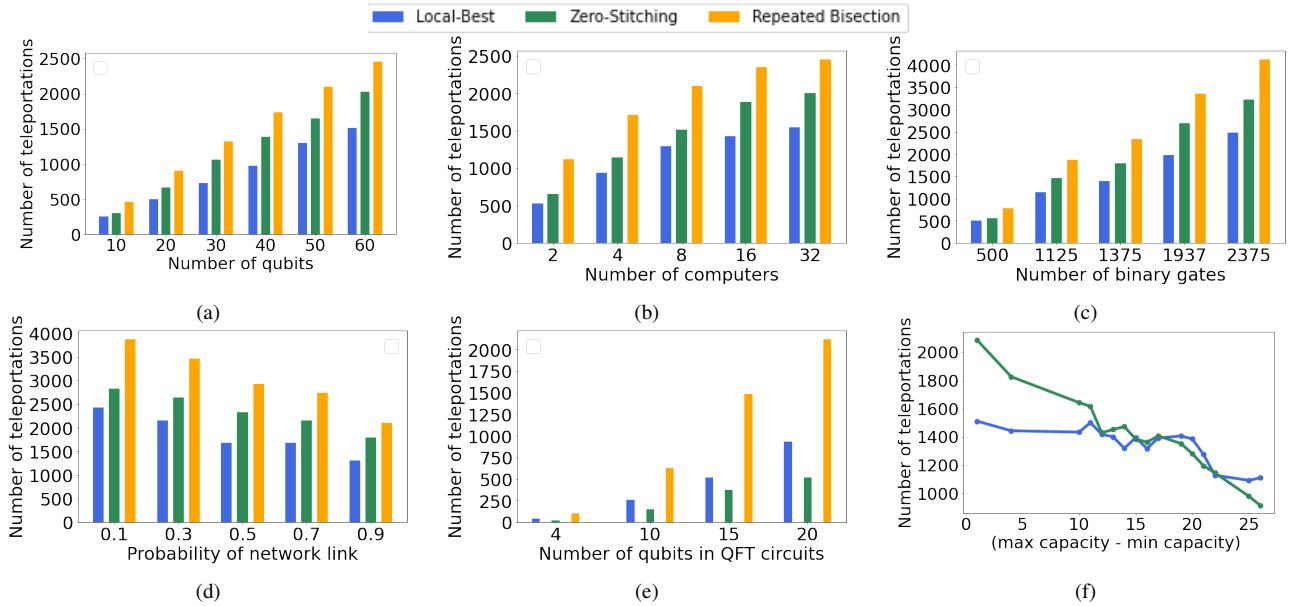

Fig. 5: Total communication cost incurred by different algorithms for varying parameter values.

problem have proposed algorithms that are exponential-time in the worst case, and hence, not compared. In particular, Nikahd et al. [18] use an integer linear program (ILP) in their algorithm that takes a prohibitively long time to run even using ILP-solvers [1] on moderately sized inputs.

Generating Random Quantum Circuits. We generate random quantum circuits using the following set of parameters:

- Varying number of qubits (default value=50)
- Varying number of gates per qubit (default value =50)
- Varying fraction f of binary gates (default value=0.5)

For given parameter values, we generate the random input circuit sequentially, choosing the the next gate to be binary with probability f. Operands of a gate are chosen randomly.

Generating Random Quantum Networks. We generate a QN using the following set of parameters (as in [12, 20]).

- Number of computers (default value=8)
- Probability of a link between a pair of nodes (default=1)

We use the Python-based library [2] to generate connected Erdős-Rényi graphs with a given edge probability. We assign values to number of vertices and probability of an edge in a way that the resulting Erdős-Rényi graph is connected with high probability. For most of the graphs used in our evaluation, we assume equal qubit storage capacities for all the nodes. This is because REPEATED BISECTION algorithm, with which our techniques are compared, is restricted to a uniform storage setting. However, we note that our techniques are general and will work for non-uniform storage capacities. In Fig. 5a-5e, we assume that the capacity of each QC is $\left\lceil \frac{n}{N_p} \right\rceil + 1$ where n is the number of qubits and N_p is the number of QCs.

Evaluation Results. We evaluate the algorithms on randomly generated circuits and networks as described above. We vary one parameter at a time, keeping the others fixed to their default values. See Fig. 5a-5f. We observe the following.

- LOCAL-BEST performs best, followed by ZERO-STITCHING, and then REPEATED BISECTION. ZERO-STITCHING offers a 30% reduction in cost compared to REPEATED BISECTION while LOCAL-BEST offers almost 50% reduction in cost.
- We see that ZERO-STITCHING performs worse than LOCAL-BEST on all randomly generated inputs presented here; we believe this is due to choosing gates uniformly at random. For benchmark circuits with repeated patterns, ZERO-STITCHING performs better (see Fig. 5e). In addition, since all of the cost in ZERO-STITCHING is in the form of stitching costs, we believe that ZERO-STITCHING's performance can be improved with a more creative algorithm for recomputing the home-computer function π_2 for C_2 in the second step of SC^*.

In Fig. 5a, the cost of all algorithms steadily increase with increasing number of qubits. In Fig. 5b, we see that the cost of LOCAL-BEST starts to plateau. In Fig. 5c, we see that the cost of each algorithm steadily increases with an increase in the number of binary gates. In Fig. 5d, the total cost decreases with an increase in the network link probability; this is because, in denser graphs, the teleportation cost between a pair of nodes is lower due to shorter distance between the nodes.

In Fig. 5f, we compare LOCAL-BEST and ZERO-STITCHING over QNs with non-uniform storage capacities. Here, the x-axis is the difference between the maximum and minimum storage capacity in the QN. Note that REPEATED BISECTION algorithm is not shown here, as it does not support such networks. Here, ZC algorithm performs better than local-best for networks with high imbalance in storage capacities.

VII. Conclusion

In this paper, we consider the problem of distributing quantum circuits over a quantum network while minimizing the total teleportation cost. In our future work, we would like to further improve the performance of ZERO-STITCHING by improving the stitching step. In addition, we would like to extend our approaches to n-ary gates.

References

[1] Cvxpy. https://www.cvxpy.org/.

[2] Networkx. https://networkx.org/documentation/stable/reference/generated/networkx.generators.random_graphs.erdos_renyi_graph.html.

[3] P. A. Martinez and C. Heunen. Automated distribution of quantum circuits via hypergraph partitioning. *Physical Review A*, 2019.

[4] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters. Teleporting an unknown quantum state via dual classical and Einstein–Podolsky–Rosen channels. *Phys. Rev. Lett.*, 70(13), 1993.

[5] O. Daei, K. Navi, and M. Z. Moghadam. Optimized quantum circuit partitioning. *ArXiv*, abs/2005.11614, 2020.

[6] Z. Davarzani, M. Z. Moghadam, M. Houshmand, and M. Nouri-baygi. A dynamic programming approach for distributing quantum circuits by bipartite graphs. *Quantum Information Processing*, 19(10):1–18, 2020.

[7] Christian L Degen, Friedemann Reinhard, and Paola Cappellaro. Quantum sensing. *Reviews of modern physics*, 89(3):035002, 2017.

[8] S. J Devitt, W. J Munro, and K. Nemoto. Quantum error correction for beginners. *Reports on Progress in Physics*, 76(7):076001, 2013.

[9] D. Dieks. Communication by EPR devices. *Physics Letters A*, 1982.

[10] G. Dósa and J. Sgall. First fit bin packing: A tight analysis. In *STACS*, 2013.

[11] J. Eisert, K. Jacobs, P. Papadopoulos, and M.B. Plenio. Optimal local implementation of non-local quantum gates. *Phys. Rev. A*, 2000.

[12] R. G. Sundaram, H. Gupta, and CR Ramakrishnan. Efficient distribution of quantum circuits. In *DISC*, 2021.

[13] M. Ghaderibaneh, H. Gupta, C.R. Ramakrishnan, and E. Luo. Pre-distribution of entanglements in quantum networks. In *IEEE QCE*, 2022.

[14] M. Ghaderibaneh, C. Zhan, H. Gupta, and C. R. Ramakrishnan. Efficient quantum network communication using optimized entanglement-swapping trees. *IEEE Transactions on Quantum Engineering*, 2022.

[15] Mark Hillery, Himanshu Gupta, and Caitao Zhan. Discrete outcome quantum sensor networks. *Physical Review A*, 107(1):012435, 2023.

[16] S. Muralidharan, L. Li, J. Kim, N. Lütkenhaus, M. D. Lukin, and L. Jiang. Optimal architectures for long distance quantum communication. *Scientific reports*, 6(1):1–10, 2016.

[17] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[18] E. Nikahd, N. Mohammadzadeh, M. Sedighi, and M. S. Zamani. Automated window-based partitioning of quantum circuits. *Physica Scripta*, 2021.

[19] J. Skorin-Kapov. Tabu search applied to the quadratic assignment problem. *ORSA Journal on computing*, 2(1):33–45, 1990.

[20] R. G. Sundaram, H. Gupta, and C. R. Ramakrishnan. Distribution of quantum circuits over general quantum networks. In *IEEE QCE*, 2022.

[21] Ranjani G Sundaram and Himanshu Gupta. Distributing quantum circuits using teleportations. *arXiv preprint arXiv:2306.00195*, 2023.

[22] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299(5886), October 1982.

[23] A. Yimsiriwattana and S. J. Lomonaco Jr. Generalized GHZ states and distributed quantum computing. *AMS Cont. Math.*, 381(131), 2005.

[24] M. Z. Moghadam, M. Houshmand, and M. Houshmand. Optimizing teleportation cost in distributed quantum circuits. *International Journal of Theoretical Physics*, 57(3):848–861, 2018.