2021 IEEE International Conference on Quantum Computing and Engineering (QCE) | 978-1-6654-1691-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/QCE52317.2021.00049

Distributing Graph States Across Quantum Networks

Alex Fischer
College of Information and Computer Sciences
University of Massachusetts, Amherst
alexander.fischer3@gmail.com

Don Towsley
College of Information and Computer Sciences
University of Massachusetts, Amherst
towsley@cs.umass.edu

***Abstract*—Graph states form an important class of multipartite entangled quantum states. We propose a new approach for distributing graph states across a quantum network. We consider a quantum network consisting of nodes—quantum computers within which local operations are free—and EPR pairs shared between nodes that can continually be generated. We prove upper bounds for our approach on the number of EPR pairs consumed, completion time, and amount of classical communication required, all of which are equal to or better than that of prior work [10]. We also reduce the problem of minimizing the completion time to distribute a graph state using our approach to a network flow problem having polynomial time complexity.**

I. Introduction

A. Motivation

Graph states form an important class of multipartite entangled states. They are interesting both theoretically, for their importance in one-way and measurement-based quantum computing [6] [13], and practically, for their applications such as to quantum metrology [15] and secure multi-party computation [7].

The role of graph states in measurement-based quantum computation makes them especially interesting as a resource to distribute across a quantum network. A classic result states that any quantum computation can be done in a "one-way" fashion [13] by preparing a graph state among a set of qubits, then performing measurements and single-qubit operations based on the measurement results. Preparing such graph states among qubits in different network nodes allows the network to perform these one-way computations in a distributed manner, which can be especially useful if different network nodes receive different parts of the input to some quantum computation. This establishes distribution of graph states across a quantum network as an important service for it to provide.

B. Prior Work

There has been considerable work on the construction of graph states at a single node [2], [9] and in the context of photonic cluster computing [1]. Much of the work on generation and distribution of graph states across a quantum network has focused on providing robustness and resilience to noise in the channels between network nodes, and memories and gates in the nodes. For example Cuquet and Calsamiglia [4] consider a similar graph state distribution protocol to ours. However, they focus on a network with a star topology as opposed to a network having an arbitrary topology as in our work. They optimize for both fidelity and fidelity decay rate given the constraint of noisy channels, instead of optimizing for EPR pair consumption and time required to distribute the graph state given the network topology.

Pirker and Dür [11] [12] consider graph state distribution protocols for more general network topologies like our work. Their focus is on their placement in a larger network protocol stack and how it can be modified to work within an unreliable network rather than their performance in terms of resource requirements and time to complete entanglement distribution.

The work of Meignant et al. [10] is closest to ours in spirit. They proposed an algorithm for constructing an *edge-decorated complete graph (EDCG)* across a network, which is then transformed into a desired graph state. They then derived upper bounds on the number of EPR pairs consumed and on time to complete the construction, under the assumption that channels, memories, and logic is perfect, and that Bell state measurements are deterministic. We conduct a similar analysis of a different algorithm to generate and distribute graph states across a network. Our approach consists of constructing the graph state at one node and transporting the qubits to the appropriate nodes within the network. Henceforth, we refer to the algorithm proposed in [10] as the EDCG algorithm.

C. Overview of Results

In this work, we study the following *Graph State Transfer (GST)* algorithm for distributing graph states across a quantum network. Suppose a set of network nodes desires to share a specific graph state, with one qubit from the graph state in each network node. The

978-1-6654-1691-7/21/$31.00 ©2021 IEEE
DOI 10.1109/QCE52317.2021.00049

idea behind our algorithm is to first create a local copy of the desired graph state at one node of the network and then distribute the graph state to the relevant set of nodes. This can be thought of as an extension of the bipartite A protocol from [4] to the general network setting. Besides introducing this generalization we make the following contributions:

- We analyze the EPR pair consumption of our GST algorithm through the derivation of an upper bound as a function of the quantum network size. We also show that the GST algorithm never consumes more EPR pairs than the EDCG algorithm. For some networks, such as those with a binary tree topology, the difference in the numbers of EPR pairs consumed can be significant.
- We derive an upper bound on the time needed to distribute the graph state (henceforth referred to as *completion time*). We present a polynomial time algorithm that chooses paths in a network that minimizes that completion time for the GST algorithm. We also show that the completion time of the GST algorithm is never more than that of the EDCG algorithm.
- We analyze the quantum memory and classical communication requirements for the GST algorithm. The memory requirements are shown to be considerably less for the GST algorithm and the classical communication overheads are comparable.

Table I details these comparisons.

Our approach to distributing graph states naturally leads to a new *resource graph state*: a graph state that can be distributed among a set of network nodes ahead of time that allows instantaneous distribution of any other graph state among those nodes by consuming the resource graph state, once that other graph state is known. This is useful if one knows the set of nodes that will request to share a graph in the future, but one does not yet know the exact graph state that will be requested. Our resource graph state requires maintaining fewer qubits than that of prior work.

II. Background

A. Graph States

A graph state [8] is a type of multiple-qubit state that is useful for certain quantum computing operations between multiple parties. We represent a graph state as a graph $G = (V, E)$ where the vertices correspond to qubits. The graph state for G is initiated with all qubits in the $|+\rangle$ state followed by the application of controlled Z operations to all pairs of qubits corresponding to pairs of vertices in E. More precisely, the graph state corresponding to G is

$$|\psi_G\rangle = \left(\prod_{(u,v)\in E} CZ_{u,v} \right) |+\rangle^{|V|}.$$

Note that CZ operations commute, so we can apply the CZ operations in any order we want (or all at once).

The graph state class of multiparty entangled states is useful because, among other reasons, there is a set of quantum operations that affect the state (up to local correction operations) graphically—ie, we can think about simple, familiar graph operations instead of quantum operators and measurements. We use the following quantum operations (and their corresponding graphical operations) in this paper:

- **Local complementation** of a vertex $a \in V$ replaces the subgraph corresponding to the neighbors of a with its complement. This operation requires $O(|N_a|)$ bits of classical communication ($O(1)$ communication between a and each of its neighbors), where N_a is the set of neighbors of a. The quantum operations required to perform this graphical operation are given in [8].
- **Edge addition/deletion** of an edge (u, v) creates an edge if one does not exist, or deletes it if it does. It corresponds to the $CZ_{u,v}$ operation.
- **Z-measurement** of a vertex a deletes a and all of its incident edges.
- **Y-measurement** of a vertex a has the effect of deleting vertex a and all of its incident edges, and locally complementing its neighbors. This operation requires $O(|N_a|)$ bits of classical communication: $O(1)$ communication between a and each of its neighbors.

A useful property of the edge addition/deletion and Y-measurement operations (along with the local correction operations implicit to Y-measurement) is that any sequence of edge addition/deletion operations and Y-measurement operations can be rewritten into an equivalent sequence of operations such that the edge additions/deletions occur first and all measurements come next. All local correction operations (one per qubit) can be executed concurrently [6] at the end. This allows us to perform a sequence of $O(n)$ edge creation and Y-measurement operations in $O(1)$ time.

B. Quantum Networks

A quantum network is a set of nodes and edges (V', E'). Nodes correspond to routers and repeaters; they are computers with unlimited numbers of qubits, the capability to perform local operations and communicate within their neighborhood in order to effect long distance entanglement. An edge represents a pair of nodes

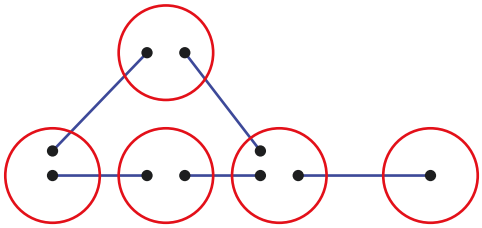


Fig. 1: An example quantum network. Red circles represent nodes; blue edges represent connections between nodes, which can be regenerated after being consumed by quantum operations within nodes.

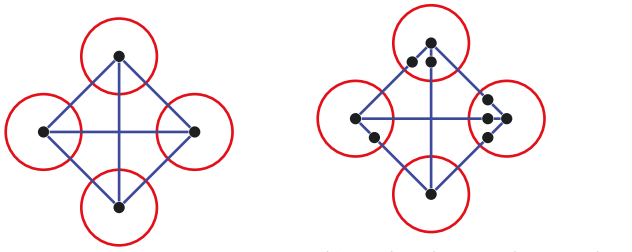
connected by a quantum channel that can generate EPR pairs between them, and can regenerate EPR pairs as necessary. The state of a quantum network at any point in time is a graph state among all the qubits in all the nodes of the network. We give an example quantum network in Figure 1.

A natural task is to distribute a graph state across a quantum network to a set of nodes. This means we alter the network state such that each qubit in a graph state exists in a specific node. Rather than preparing a graph state from some other graph state among a specified set of qubits via graphical operations, which is not always possible (in fact, it is NP-complete to determine whether transforming one graph state into another is possible [5]), we only require that each qubit in the graph state be part of a specified node. To achieve this, we can use local operations within nodes (which are free in our model), and link-level EPR pair regeneration. EPR pair regeneration, an operation on qubits in different nodes, is expensive—EPR pair consumption is one of the performance metrics of a quantum networking algorithm in this model.

For the problem of distributing an arbitrary graph state among a network with n nodes, Meignant et al. [10] give an algorithm that consumes at most $\frac{n(n-1)}{2}$ EPR pairs and has a completion time of at most $n-1$ timesteps. They also propose a "resource graph state" (see Figure 2) that can be distributed among a network ahead of time in order to enable instantaneous distribution of an arbitrary graph state. Their resource graph state requires $\frac{n(n-1)}{2}$ qubits. We present a graph state distribution algorithm that uses at most $\frac{n(3n/2-1)}{2}$ EPR pairs. This algorithm naturally leads to an alternate resource graph state that requires $2(n-1)$ qubits.

III. Connection Transfer

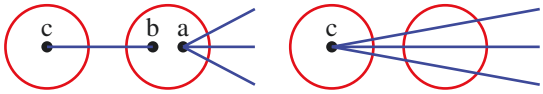
We start with a simple sequence of operations we refer to as *connection transfer*. This operation starts with a qubit a at a node $A \in V'$ that is connected to other qubits, which are possibly outside A. A also includes



(a) Complete graph

(b) Edge-decorated complete graph (EDCG)

Fig. 2: A 4 node example of a resource graph state. (a) A 4-node network such that the network graph state is the complete graph among 4 qubits, each qubit in a different node. (b) A 4-node network such that the network graph state is the *edge-decorated complete graph*: a complete graph with additional vertices added to split each edge into two. The additional vertices added can exist in either of the nodes of the edge which that vertex split in two. Z or Y measuring a decoration vertex deletes or preserves the original edge, respectively. By Z or Y measuring each decoration vertex (along with associated local correction operations), any 4-qubit graph state can be prepared among the 4 nodes.



(a) Setup.

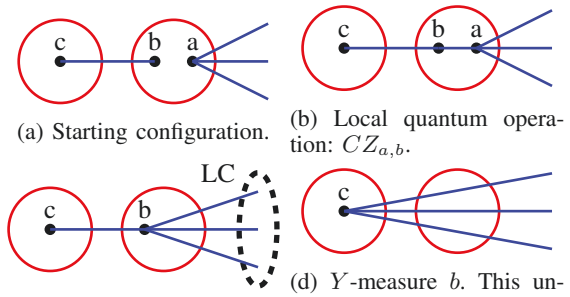
(b) End result.

Fig. 3: The setup and end result of the connection transfer process. We transfer the edges connected to a to qubit c, by consuming the EPR pair between b and c.

a second qubit b that is entangled with a third qubit c residing at another node. Connection transfer changes the network graph state such that the edges between qubit a and its neighborhood are connected to qubit c instead of a. See Figure 3 for the setup and end result of connection transfer. We present two approaches to connection transfer: via graphical operations, and via teleportation.

Figure 4 details connection transfer via graphical operations. First we create an edge between a and b with a local CZ operation. Then we Y-measure both a and b. The successive Y-measurements locally complement a's neighborhood twice, but the second such local complementation undoes the first, making the net effect of the two Y-measurements to transfer a's connections to c. This process consumes one (non-local) EPR pair.

Connection transfer via teleportation is straightforward. Again, we start with a qubit a whose edges we wish to transfer to a qubit c. Qubit c is connected to a qubit b located in the same node as a. This situation is



(a) Starting configuration.

(b) Local quantum operation: $CZ_{a,b}$.

(c) Y-measure a. Results in locally complementing a's (former) neighborhood.

(d) Y-measure b. This undoes the local complementation of the neighborhood of a.

Fig. 4: Connection transfer via graphical operations.

depicted in Figure 3(a). The initial state is

$$|\psi_G\rangle = \frac{1}{\sqrt{2}}\left(|+\rangle_b|0\rangle_c + |-\rangle_b|1\rangle_c\right) \frac{1}{\sqrt{2}}\left(|0\rangle_a + |1\rangle_a \prod_{v\in N_a} Z_v\right) \left(\prod_{(u,v)\in E''} CZ_{u,v}\right)|+\rangle^{\otimes|V\setminus\{a,b,c\}|}$$

where E'' is the edge set of the network's graph state except for those incident to a, and except for (b,c). We break this expression down term by term. The $|+\rangle^{\otimes|V\setminus\{a,b,c\}|}$ term corresponds to all qubits except a, b, and c prepared in the $|+\rangle$ state. The $\prod_{(u,v)\in E''} CZ_{u,v}$ operations create all the edges except for (b,c) and those connected to a. The $\frac{1}{\sqrt{2}}\left(|0\rangle_a + |1\rangle_a \prod_{v\in N_a} Z_v\right)$ term creates the qubit a and the edges between a and the vertices in its neighborhood N_a. The $\frac{1}{\sqrt{2}}\left(|+\rangle_b|0\rangle_c + |-\rangle_b|1\rangle_c\right)$ term creates the qubits b and c and the edge between them.

It is easy to see that measuring qubits a and b in the basis

$$\left\{\begin{array}{c}\frac{1}{\sqrt{2}}\left(|0\rangle_a|+\rangle_b \pm |1\rangle_a|-\rangle_b\right), \\ \frac{1}{\sqrt{2}}\left(|0\rangle_a|-\rangle_b \pm |1\rangle_a|+\rangle_b\right)\end{array}\right\} \quad (1)$$

results in the desired transfer of a's connections to c. To see this, consider what happens when we obtain the measurement result $|\phi\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_a|+\rangle_b + |1\rangle_a|-\rangle_b\right)$:

$$\langle\phi|\psi_G\rangle = \frac{1}{2\sqrt{2}}\left(|0\rangle_c + |1\rangle_c \prod_{v\in N_a} Z_v\right) \left(\prod_{(u,v)\in E'} CZ_{u,v}\right)|+\rangle^{\otimes|V\setminus\{a,b,c\}|}.$$

This is precisely the graph state depicted in Figure 3(b). If the measurement result is another Bell state besides $|\phi\rangle$ then an X and/or Z gate correction will also need to be applied to qubit c, as is done with conventional quantum teleportation of one qubit.

Note that the graphical and teleportation approaches to connection transfer are essentially equivalent—the local CZ operation of the graphical approach effects a change of basis, allowing the 2 single-qubit Y-measurements to achieve the same effect as the multi-qubit measurement in the basis (1) used in teleportation. The graphical approach requirement that each node be able to perform local CZ operations and Y-measurements is no stricter than the operation requirements of nodes considered in prior work [10].

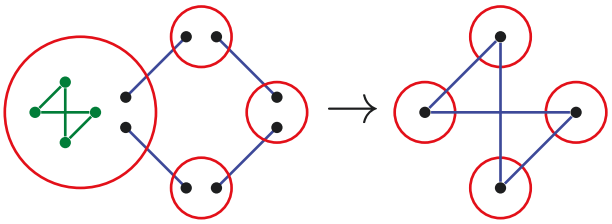


Fig. 5: Example setup and end result of our GST algorithm. A local copy of the final graph state (green) is prepared within a node and distributed throughout the network.

IV. Graph State Distribution

We can transfer a qubit's connections to a node not connected to that qubit's node through a sequence of connection transfers along a path of edges in the network running from the starting node to the desired node. This suggests the following algorithm for generating a graph state. First, generate a local copy of the graph state at some node via local CZ operations, which are free in our model. Next, transfer the connections (using either of the aforementioned connection transfer methods) of each qubit to its corresponding node. Figure 5 illustrates the starting state and end result of this algorithm for an example network and desired final graph state. We call this algorithm the Graph State Transfer algorithm, or the **GST algorithm**.

This requires the root node to maintain $|S|$ qubits (where S is the set of network nodes that will share the final graph state) and prepare them in an entangled state via local CZ operations. This may be a difficult requirement to meet for large networks; however, it is also required of the graph state distribution approach in [10].

A. Resource Graph State

This approach to distributing graph states suggests a resource graph state—a graph state that can be distributed among a set of nodes ahead of time that allows any arbitrary graph state to be distributed among those

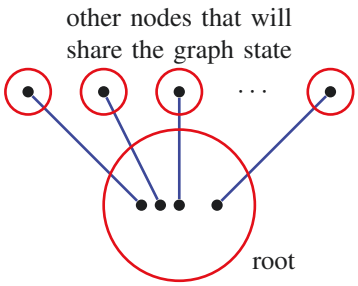


Fig. 6: A resource graph state that requires $2(n-1)$ qubits, where n is number of nodes that will share the final graph state.

nodes in one timestep. Resource graph states are useful if we know ahead of time that a set of nodes (or a superset of nodes) will request a graph state, but we do not know what that graph state will be. We choose one node in the network (called the "root node") and have the network graph state be such that the root node shares an entangled pair with every other node that will share the desired final graph state, as in Figure 6. This allows us to generate an arbitrary graph state in one time step by generating the local copy at the root and distributing the graph state as usual, using either connection transfer method.

This resource graph state requires $2(n-1)$ qubits, where n is the number of nodes that will share the graph state. This is an improvement over the $\frac{n(n-1)}{2}$ qubits needed for the EDCG (Figure 2), the resource graph state proposed in [10]. This improvement is significant because long-term maintenance of memory qubits is, and likely will continue to be, a challenging engineering problem.

V. EPR Pair Consumption

Each connection transfer operation consumes one EPR pair. For each qubit in a graph state whose connections are transferred to a relevant node, those connection transfers consume a number of EPR pairs equal to the length of the path in the network from the root node the relevant node. Thus the total number of EPR pairs used to distribute a graph state across a network depends on the choice of paths from the root node to every other node in the network (and also implicitly depends on the choice of a root node). The number of EPR pairs consumed equals the sum of the lengths of such paths.

Upper bounding the number of EPR pairs consumed by this algorithm when distributing a graph state among a set of nodes S is thus equivalent to upper bounding the sum of minimum path lengths from some root node to every node in S. We upper bound this sum by choosing the root node to our advantage. For any connected graph with n vertices and any vertex v, at most $n-i$ vertices can be distance i away from v. This means the sum of minimum path lengths from v to all vertices in S, which we call $Ne(S)$, is at most

$$\begin{aligned} N_e(S) &\leq (n-1)+(n-2)+\cdots+(n-|S|) \\ &= \frac{|S|(2n-|S|-1)}{2}. \end{aligned}$$

In particular, if $S = V'$ (ie. we are distributing a graph state across the entire network) then we use at most

$$\begin{aligned} N_e(V') &\leq (n-1)+(n-2)+\cdots+1 \\ &= \frac{n(n-1)}{2} \end{aligned} \tag{2}$$

EPR pairs. Note that this upper bound is achieved when the network is a linear graph with the root at one end of the line.

Suppose we have the flexibility to select any vertex as the root. For any connected graph with n vertices and maximum degree of at least two, basic graph theory tells us that there exists a vertex v that is distance at most $\lceil \frac{n-1}{2} \rceil$ from any other vertex (see eg. [14] Theorem 4.1). Thus by choosing a root in the center of the graph, we can replace every term in the above sum that is at least $\lceil \frac{n-1}{2} \rceil$ by $\lceil \frac{n-1}{2} \rceil$ to get a better upper bound. We compute this bound for even and odd n.

For even n, we replace every term in the sum from (2) that is at least $\frac{n}{2}$ by $\frac{n}{2}$ to get

$$\begin{aligned} N_e(V') &\leq \frac{n}{2}\cdot\frac{n}{2} + \left(\frac{n}{2}-1\right) + \left(\frac{n}{2}-2\right) + \cdots + 1 \\ &= \frac{n^2}{4} + \frac{\frac{n}{2}\left(\frac{n}{2}-1\right)}{2} \\ &= \frac{3n^2-2n}{8}. \end{aligned} \tag{3}$$

For odd n, we replace every term in the sum from (2) that is at least $\frac{n-1}{2}$ by $\frac{n-1}{2}$ to get

$$\begin{aligned} N_e(V') &\leq \frac{n+1}{2}\cdot\frac{n-1}{2} + \left(\frac{n-1}{2}-1\right) \\ &\quad + \left(\frac{n-1}{2}-2\right) + \cdots + 1 \\ &= \frac{n^2-1}{4} + \frac{\frac{n-1}{2}\left(\frac{n-1}{2}-1\right)}{2} \\ &= \frac{3n^2-4n+1}{8}. \end{aligned} \tag{4}$$

As $n > 0$, the even n bound from (3) is greater than the odd n bound from (4), so in general $N_e(V') \leq (3n^2 - 2n)/8$.

This EPR pair consumption upper bound is lower than that for the EDCG algorithm, [10], which uses up to $\frac{n(n-1)}{2}$ EPR pairs. However, we can also prove a stronger result: that we always use less than or an equal number of EPR pairs used by the EDCG algorithm.

The EDCG algorithm creates an edge-decorated complete graph (EDCG) between the set of nodes S that

will share the final graph state (see Figure 2). To create an EDCG among $S = \{s_1, s_2, \cdots, s_m\}$, the EDCG algorithm creates an m-qubit GHZ state between $\{s_1, \cdots, s_m\}$, then an $m-1$ qubit GHZ state between $\{s_2, \cdots, s_m\}$, then an $m-2$ qubit GHZ state between $\{s_3, \cdots, s_m\}$, etc, until creating a 2 qubit GHZ state (ie, just an EPR pair/edge in the network's graph state) between s_{m-1} and s_m. These GHZ states are then combined via local operations at each node to form an EDCG. Then, each edge in the complete graph is either deleted or kept, by Z-measuring or Y-measuring the decoration vertex on that edge, respectively. See [10] Figures 7 and 8 for more detail.

Theorem 1. *The GST algorithm always uses less than or an equal number of EPR pairs used by the EDCG algorithm.*

Proof. Creating a GHZ state between $\{s_k, \cdots, s_m\}$ requires at least as many EPR pairs as performing connection transfer from s_m to s_k, as the former will use EPR pairs along a path in the network between s_k and s_m (and possibly more EPR pairs). Thus, performing connection transfer $|S| - 1$ times between s_m and the other nodes in S uses no more EPR pairs than generating all the GHZ states required by the EDCG algorithm. So by choosing s_m as our root node, we use no more EPR pairs than the EDCG algorithm does to distribute a graph state among nodes in S. □

A. An Example: Full Binary Tree

Here we provide an example where there is a large gap in the numbers of EPR pairs consumed by the GST algorithm and by the EDCG algorithm to distribute a graph state among every node in a full binary tree. Consider a full binary tree of height h (by convention we consider the trivial tree with 1 vertex to have height 0); this graph has $n = 2^{h+1} - 1$ vertices. We choose the root of the tree as the root node of the network, and the paths we use to transfer connections to every other node of the network are obvious as there is only one choice of path for each node since the network structure is a tree. The number of EPR pairs consumed is the sum of path lengths from the root node to every other node:

$$\sum_{i=1}^{h} i2^i = 2^{h+1}(h-1) + 2 = \Theta(n \log n).$$

It is a straightforward calculation to show that the EDCG algorithm requires $\frac{n(n-1)}{2}$ EPR pairs to distribute a graph state to every node in this network with the EDCG algorithm. For this example, the EDCG algorithm consumes $\Theta(n/\log n)$ more EPR pairs.

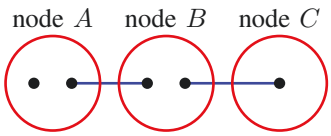


Fig. 7: The setup for teleportation along two edges in the network. The pair of qubits in node A and the pair of qubits in node B will be measured in the basis in Equation (1).

VI. Minimizing Completion Time

A. Parallelization

First, we show that any sequence of connection transfers that does not require using any network edge more than once can be done simultaneously in one timestep.

For the graphical connection transfer approach, note that any sequence of connection transfers is a sequence of controlled-Z operations, Y measurements (at most one per qubit), and local correction operations required by the measurement results. We can rearrange those operations [6] (see Section 5.2 of [6] for details) into a different sequence of operations with the same effect such that the new sequence of operations consists first of controlled-Z operations, followed by measurements, and then local correction operations. The controlled-Z operations will be done on the same qubit pairs as the original sequence of operations, and the measurements and local correction operations will also be done on the same qubits as the original sequence.

This means any sequence of connection transfer operations that does not use any edge in the network more than once can equivalently be done as a sequence of (in order):

1) Controlled-Z operations. These can all be done at once as these operations commute.
2) Measurements. These can all be done at once because the measurements are of different qubits.
3) Local correction operations based on the measurement results.

The teleportation approach to connection transfer, like the graphical approach, also allows connection transfers among distinct edges in the network to be parallelized. We can perform the necessary local correction operations (X and/or Z gates) on the final qubit in the connection transfer path based on the results of all the measurements in the connection transfer path. The exact correction operations on the final qubit are the correction operations that would have been done on the qubits in the path, done in reverse order of their appearance in the path.

To illustrate this rearrangement of operations from multiple teleportations, we show exactly how it works for the case of two teleportations in a row; see Figure

7 for the setup. When teleporting a state from node A to node B and then from node B to node C, the usual sequence of operations is:

1) Measure two qubits in node A in the basis in Equation (1).
2) Based on the measurement result, perform a correction operation (X and/or Z gates) on a qubit in node B.
3) Measure two qubits in node B in the basis in Equation (1).
4) Based on the measurement result, perform a correction operation (X and/or Z gates) on a qubit in node C.

Note that instead of correcting for the first measurement result in step 2, we can teleport the uncorrected state from node B to node C and then perform the correction operations that we would have done in step 2 on the qubit in node C. This results in the sequence of operations:

1) Measure two qubits in node A in the basis in Equation (1).
2) Measure two qubits in node B in the basis in Equation (1).
3) Based on the second measurement result, perform a correction operation (X and/or Z gates) on the qubit in node C.
4) Based on the first measurement result, perform a correction operation (X and/or Z gates) on the qubit in node C.

We can do both measurement operations at once since they are performed on different qubits. The measurement results can then be reported to node c, followed by all local correction operations on node c.

This also easily generalizes to sequences of more than two teleportations; we just continue teleporting uncorrected states to the last qubit in the path and then do all correction operations at that last qubit. This means the only operations done at each node in a path of teleportation connection transfers (except the last node in the path) are the measurement operations, which can all be done at once because they are measurements on different qubits. Also, because the only local correction operations are performed at the end of any chain of connection transfers, the measurement results need only be communicated to the last node of any path.

Even though the final qubit in a chain of n teleportations may require up to $2n$ correction operations, that sequence of operations will be equivalent to one of the 16 elements of the Pauli group on 1 qubit. So the up to $2n$ correction operations required can be accomplished in $O(1)$ time by applying the relevant element from the Pauli group.

We refer to the time it takes to perform simultaneous CZ operations, measurements, local correction operations, and generate any EPR pairs as needed, as a timestep. Thus any sequence of connection transfer operations that does not use any edge in the network more than once, done via the graphical approach or teleportation approach, can be executed in one timestep. In general, distributing a graph state across an n node network will take no more than $n-1$ timesteps. This is because the connection transfers on each path from the root node to another node can be executed in one timestep, and there are at most $n-1$ such paths—one per node that receives connections from the root node.

B. Optimization via Path Selection

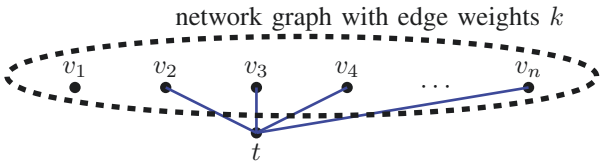
We can often do better than $n-1$ timesteps. We can minimize the completion time by solving a network flow problem (see eg. [3] chapter 26). Specifically, given a network graph, a root node, and a set of vertices S of the network graph that will share the final graph state, we construct a network flow problem instance such that its maximum flow is $|S|$ iff there is a set of $|S|$ paths from the root to the vertices in S such that no edge in the graph is used more than k times (which allows us to distribute a graph state in k timesteps). A binary search on k, as well as trying all possible root nodes, gives the optimal time to distribute a graph state among the nodes in S.

The construction is as follows. Start with the original network graph with each edge having weight k. Add a new vertex t. Finally, add edges from each vertex in S to t with weight 1 (see Figure 8). This network flow problem instance is related to completion time minimization by the following theorem.

Theorem 2. *In the network flow construction given in Figure 8, the max flow from the root node to t is $|S|$ iff there exist $|S|$ paths in the network, each from the root to a different node in S, such that each edge is used at most k times.*

Proof. The $\Longleftarrow$ direction is obvious, as we can construct a flow of value $|S|$ by adding all of the $|S|$ paths, and setting the flow of the edges from S to t to be one.

For the $\Longrightarrow$ direction, start with a maximum flow of value $|S|$ from the root node to t. The flow decomposition theorem allows us to decompose a max flow of value $|S|$ into path flows (and cycle flows, which we can ignore) that combine to form the max flow. Because there are $|S|$ edges going into t each with weight one, those path flows must have value one and there must be $|S|$ of them. Those $|S|$ path flows each with value one from the root node to t give us $|S|$ paths from the root node to each node in S. Because each edge in the network flow instance has capacity at most k, each edge



(edges from S to t have weight 1)

Fig. 8: Let every edge in this graph (which is the network graph, plus $|S|$ edges from every node in S to an extra vertex t) have weight k, except for the edges to t which have weight 1. Then the max flow from v_1 (the root) to t is $|S|$ iff there is set of $|S|$ paths from the root to every node in S such that every edge in the network is used at most k times.

in the original network graph must be used at most k times by all the paths. □

Our completion time minimization algorithm is given in Algorithm 1. See Figure 9 for an example of connection transfer paths found by our network flow approach.

Note that selecting connection transfer paths in the network that minimize completion time may result in more EPR pairs consumed than indicated by our previously derived upper bound. This is because the paths found from the network flow problem may not be the shortest paths from the root node to the nodes in S, and the EPR pair consumption bound relies on using the shortest paths in the network. Exploring the tradeoff between reducing completion time and reducing EPR pair consumption is a subject for future work.

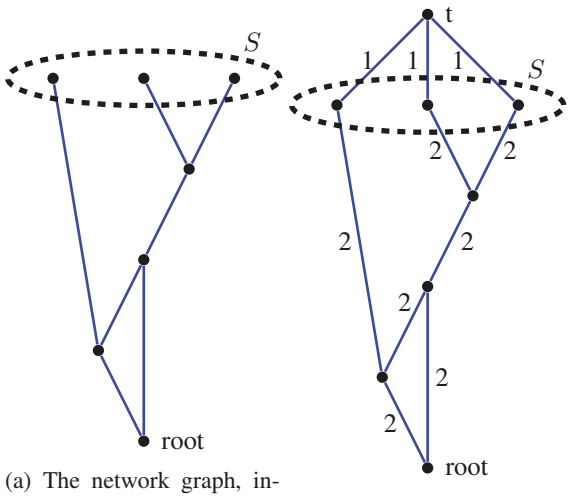
C. An Example: Full Binary Tree

In Section V.a, we computed the number of EPR pairs consumed when distributing a graph state among every node of a network with a full binary tree structure. If we use the same root node (the root of the tree) and paths as we introduced in Section V.a, then the two edges connected to the root node will each be used $\frac{n-1}{2}$ times, and every other edge in the network will be used fewer times. Thus the completion time for the GST algorithm to distribute a graph state to every node of that network is $\frac{n-1}{2}$ timesteps. This is an improvement over the $n-1$ timesteps required by the EDCG algorithm.

VII. Classical Communication Requirements

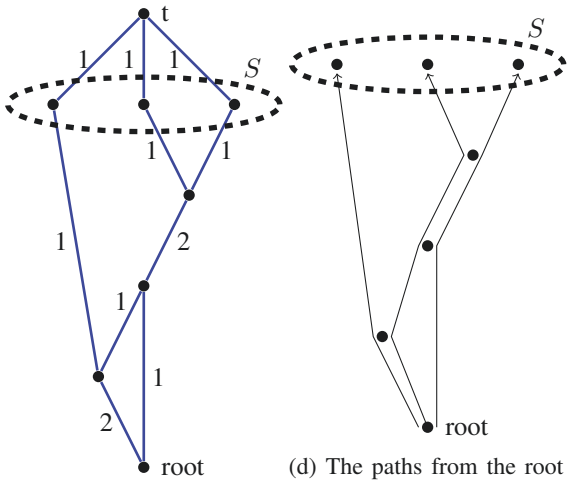
Both graphical and teleportation based graph state distribution require $O(n^2)$ bits of classical communication to distribute a graph state across a network of size n. The classical communication requirement comes from communicating measurement results so nodes can perform the appropriate local correction operations.

For the graphical approach—if, for each qubit whose connections we transfer to its destination node, we



(a) The network graph, including the root and S.

(b) The network flow problem instance with $k = 2$.



(c) A maximum flow from the root to t, with value $3 = |S|$. All flow directions are upward.

(d) The paths from the root to S found by the flow decomposition theorem. Each edge is used at most $k = 2$ times.

Fig. 9: An example of connection transfer paths found by our network flow approach.

reorder the edge addition, Y-measurement, and local correction operations such that the local corrections come last [6] (see Section VI for details on this process), then we send $O\left(n^2\right)$ classical bits for measurement results. This is because each time we transfer the connections of a qubit to its destination node, we can transmit all $O(n)$ measurement results to the root node, which will then transmit all $O(n)$ correction operation requirements (which require $O(1)$ classical communication each) to the nodes which require local correction. Thus we require $O(n)$ classical communication for each qubit whose connections we transfer to a destination node, or $O\left(n^2\right)$ communication total.

Algorithm 1 Our algorithm for minimizing the completion time of distributing a graph state in a network with graph structure G, to some subset S of nodes in the network.

‘

procedure NETWORKFLOW(G, S, k, root)
    Start with the network graph G and assign all edge weights k.
    Add a vertex t.
    Add $|S|$ edges, from every vertex in S to t, with weight 1.
    Let root be the source node of this network flow instance and let t be the sink node.
    return the max flow of this network.
procedure MINIMIZECOMPLETIONTIME(G, S)
    for all possible root nodes $v \in V(G)$ **do**
        Use binary search on k to find the minimum $k \in \{1, 2, \cdots, |S|\}$ such that $\mathrm{NetworkFlow}(G, S, k, v)$ has max flow $|S|$.
        Let k_v be this minimum k value.
    Let $\mathrm{root} = \arg\min_{v \in V(G)} k_v$.
    Let $k = \min_{v \in V(G)} k_v$.
    Use the flow decomposition theorem to extract $|S|$ paths from $\mathrm{NetworkFlow}(G, S, k, \mathrm{root})$.
    Use these $|S|$ paths to distribute the graph state among the network from the root node to the nodes in S.

For the teleportation approach—when transferring any qubit's connections to its destination node, the local correction operation at the destination node depends on the measurement results of all of the $O(n)$ measurements done at each node along the path in the network. Hence each node that shares the final graph state requires $O(n)$ qubits of classical communication, for a total of $O(n^2)$ bits of classical communication.

The EDCG algorithm for graph state distribution also requires $O(n^2)$ classical communication. A star expansion operation on a qubit with m neighbors requires $O(m)$ bits of classical communication. Distributing a GHZ state across a graph with n vertices requires that each node only be communicated with once, so only $O(n)$ bits of communication are required. The EDCG algorithm requires distributing a GHZ state n times, so $O(n^2)$ bits of classical communication are required. Also, performing edge measurements and subsequent local corrections to turn the EDCG state into the desired graph state requires $O(1)$ bits for each edge, for $O(n^2)$ bits total.

Note that both of our connection transfer approaches result in $O(1)$ bits of classical communication required for each measurement done—equivalently, $O(1)$ bits of classical communication for each EPR pair consumed. Also, the EDCG algorithm requires $\Omega(1)$ bits of classical communication per EPR pair consumed as that also requires communicating measurement results of each measurement that consumes an EPR pair. Thus Theorem 1 from Section V also extends from EPR pair consumption to bits of classical communication—if the EDCG algorithm for graph state distribution uses $c(n)$ bits of classical communication to distribute a graph state in a network of size n, then the GST algorithm uses $O(c(n))$ bits.

Table I summarizes all the cost metrics of our graph state distribution algorithm compared to the EDCG algorithm [10].

ACKNOWLEDGMENTS

We would like to acknowledge Professor Neil Immerman for coming up with the clever network flow construction. This work was supported in part by the National Science Foundation (NSF) under Grants CNS-195744 and ERC-1941583.

REFERENCES

[1] H. J. Briegel, Browne D. E., W. D urer, R. Raussendor, and M. Van den Nest. Measurement-based quantum computation. *Nature Phys.*, 5(6):19–26, 2009.
[2] Earl T Campbell. Distributed quantum-information processing with minimal local resources. *Physical Review A*, 76(4):040302, 2007.
[3] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.
[4] Martí Cuquet and John Calsamiglia. Growth of graph states in quantum networks. *Phys. Rev. A*, 86:042304, Oct 2012.
[5] Axel Dahlberg, Jonas Helsen, and Stephanie Wehner. How to transform graph states using single-qubit operations: computational complexity and algorithms. *arXiv preprint arXiv:1805.05306*, 2018.
[6] Vincent Danos, Elham Kashefi, and Prakash Panangaden. The measurement calculus. *Journal of the ACM (JACM)*, 54(2):8–es, 2007.
[7] Marc Hein, Wolfgang Dür, Jens Eisert, Robert Raussendorf, M Nest, and H-J Briegel. Entanglement in graph states and its applications. *arXiv preprint quant-ph/0602096*, 2006.
[8] Marc Hein, Jens Eisert, and Hans J Briegel. Multiparty entanglement in graph states. *Physical Review A*, 69(6):062311, 2004.
[9] Yuichiro Matsuzaki, Simon C Benjamin, and Joseph Fitzsimons. Probabilistic growth of large entangled states with low error accumulation. *Physical review letters*, 104(5):050501, 2010.

	EPR pairs	Time	Res. Graph State Qubits	Classical Comm.	Bin. Tree EPR Pairs	Bin. Tree Completion Time
GST algorithm	$\frac{3n^2-2n}{8}$	$n-1$	$2(n-1)$	$O\left(n^2\right)$	$2^{h+1}(h-1)+2$ $=\Theta(n\log n)$	$\frac{n-1}{2}$
EDCG algorithm	$\frac{n(n-1)}{2}$	$n-1$	$\frac{n(n+1)}{2}$	$O\left(n^2\right)$	$\frac{n(n-1)}{2}$	$n-1$

TABLE I: Comparison of various performance metrics between the GST algorithm and the EDCG algorithm. We also include a comparison of how both algorithms perform on the binary tree of height h example given in Section V.a.

[10] Clément Meignant, Damian Markham, and Frédéric Grosshans. Distributing graph states over arbitrary quantum networks. *Phys. Rev. A*, 100:052333, Nov 2019.

[11] A Pirker, J Wallnöfer, and W Dür. Modular architectures for quantum networks. *New Journal of Physics*, 20(5):053054, may 2018.

[12] Alexander Pirker and Wolfgang Dür. A quantum network stack and protocols for reliable entanglement-based networks. *New Journal of Physics*, 21(3):033003, 2019.

[13] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86:5188–5191, May 2001.

[14] Roswitha Rissner and Rainer E. Burkard. Bounds on the radius and status of graphs. *Networks*, 64(2):76–83, 2014.

[15] Nathan Shettell and Damian Markham. Graph states as a resource for quantum metrology. *Physical Review Letters*, 124(11):110502, 2020.