Greedy Algorithms for Finding Entanglement Swap Paths in Quantum Networks

Anoop Kumar Pandey
Computer Science and Engineering
IIT Hyderabad
Hyderabad, Telangana, India
cs21resch04002@iith.ac.in

Anubhav Srivastava
Computer Science and Engineering
IIT Hyderabad
Hyderabad, Telangana, India
cs21mtech02001@iith.ac.in

Shuhul Handoo
EECS
IISER Bhopal
Bhopal, Madhya Pradesh, India
shuhul18@iiserb.ac.in

Bheemarjuna Reddy Tamma
Computer Science and Engineering
IIT Hyderabad
Hyderabad, Telangana, India
tbr@cse.iith.ac.in

MV Panduranga Rao
Computer Science and Engineering
IIT Hyderabad
Hyderabad, Telangana, India
mvp@cse.iith.ac.in

ABSTRACT

The entanglement swap primitive facilitates the establishment of shared entanglement between non-adjacent nodes in a quantum network. This shared entanglement can subsequently be used for executing quantum communication protocols. The fundamental problem in quantum networks is to determine a path for entanglement swapping in response to *demands* for entanglement sharing between pairs of nodes. We investigate variants of this problem in this work.

We propose a framework of Greedy algorithms that can be tweaked towards optimizing on various objective functions. In conjunction with a novel Spatial and Temporal (split across multiple paths) splitting approach to entanglement routing, we use this framework, which we call GST, to investigate the scenario when the demands are specified in terms of a starting time and a deadline. Considering the fragile nature of quantum memory, "bursty" demands are natural, and therefore the setting is important.

We study the algorithm for maximizing the number of satisfied demands and the number of entangled pairs shared. We report empirical results on the performance against these objective functions, and compare with a naive algorithm that involves neither temporal and spatial splitting of the demands, nor the greedy approach to scheduling the demands.

ACM Reference Format:
Anoop Kumar Pandey, Anubhav Srivastava, Shuhul Handoo, Bheemarjuna Reddy Tamma, and MV Panduranga Rao. 2023. Greedy Algorithms for Finding Entanglement Swap Paths in Quantum Networks. In *24th International Conference on Distributed Computing and Networking (ICDCN 2023), January 4–7, 2023, Kharagpur, India.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3571306.3571408

1 INTRODUCTION

Quantum Networks are next-generation networks used to communicate quantum information across quantum nodes [1]. An important primitive for quantum communications is entanglement sharing–the sharing of entangled particles between two quantum nodes. Once an entangled pair is shared, interesting protocols like quantum teleportation can be accomplished.

For sharing EPR[1] pairs between "distant" quantum nodes, quantum repeaters are used, which employ "entanglement swapping" protocols to achieve the objective [2, 3]. In this work, we assume a quantum network with quantum nodes that can generate demands for EPR sharing as well as act as repeaters. The presence of an edge between two quantum nodes indicates the presence of a direct (optical fiber-based) link for quantum and classical communications. EPR pairs can be trivially shared over such links. The problem, at a high level, is to find a path between two nodes on the quantum network, on which the entanglement swap protocol can take place. On successful execution of the protocol, EPR pairs are shared between the two nodes. We discuss some of the ideas that are prerequisite to our work, in Section 2.1.

A significant body of work exists for finding entanglement swap paths that optimize on various parameters like path length, fidelity and time bounds (please see Section 2 for a quick review). In this paper, we study a simple yet useful variant of the EPR sharing problem. The lifetime of quantum memory elements (coherence times) is very low. For applications that require (say) local unitary operations on multiple EPR pairs at one node, it is essential to have all the EPR pairs shared within a short span of each other, as a *burst*. In other words, a "demand" to share a given number of EPR pairs between two nodes i and j is also accompanied by a start time $s_{i,j}$ and a deadline $e_{i,j}$, $s_{i,j} < e_{i,j}$, during which all the EPR pairs have to be shared. To the best of our knowledge, there has been no attempt to maximize the number of satisfied demands under the burst condition.

We treat the quantum network as a resource for which demands are submitted for an operational time epoch of length T time units ($[0, T-1]$). Demands for EPR exchange are submitted in advance and they scheduled to be serviced in the operational time epoch of

[1]A maximally entangled pair, named after Einstein, Podolsky and Rosen.

$[0, T-1]$. This "static" picture of the quantum network as a resource is particularly relevant when the nodes are processing elements of the same quantum system.

We design a greedy heuristic framework that can be used for different objective functions. In the context of the burst application, we apply the framework to maximize the number of demands satisfied, and the number of EPR pairs shared overall. The framework is a generic one, and can be instantiated to yield different algorithms towards optimizing for different parameters.

While earlier works exist that split the demands among multiple paths, a novel feature of our approach is that we split the demands both in terms of time and across multiple paths whenever possible, while respecting the burst constraints.

The paper is arranged as follows. Section 2.2 briefly reviews existing literature, particularly with the objective of comparing and contrasting with our approach. We present the Greedy framework with Spatial and Temporal splitting (GST) in Section 3. In Section 4, we discuss empirical results wherein we compare the GST approach against a naive approach that simply processes the demands in the order in which they are presented. Finally, we conclude the paper in Section 5.

2 BACKGROUND

In this section, we will first discuss some basic theory of quantum networks required for reading this paper, followed by a brief survey of related work in the area. We then present the formal problem statement addressed in this paper.

2.1 Quantum Networks Basics

We begin with a very brief discussion of quantum communications prerequisites. For details, please see the classic text by Nielsen and Chuang [4]. A quantum state is described by a unit norm vector in a Hilbert space. The quantum analogue of a bit, the so-called *qubit*, is such a vector in a two-dimensional Hilbert space. *Entangled* qubits are a very useful and interesting resource in quantum computing and communications. We will be interested in pairs of such "maximally" entangled qubits, called EPR pairs. Described by the state vector $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$, this system is central to important quantum communication protocols like teleportation.

For such protocols to execute between two quantum nodes, one particle in each of the EPR pairs has to be shared between them. The distribution of EPR pairs is easiest if the quantum nodes are adjacent to each other in the network. If separated by multiple edges, then an entanglement swapping protocol accomplishes the task of EPR distribution. Consider quantum nodes a, b and c, with an edge (that is, a fiber optic link) between a and b, and b and c. Entanglement swapping converts EPR pairs between a and b, and b and c to a shared EPR between a and c. Thus there are two key operations in entanglement swapping: (i) entanglement generation (and distribution between adjacent nodes) and (ii) the so-called Bell State Measurement that results in the above mentioned conversion [2, 3].

This protocol can be lifted to longer paths in the quantum network. It is easy to see that $O(\log n)$ entanglement swap operations are required to distribute EPR pairs between the end nodes of a path of length n. [2]

A central problem in quantum networks is to identify the best path along which entanglement swapping can be carried out, given a demand for a number EPR pairs to be established between two nodes in the quantum network.

2.2 Previous Works in Quantum Networks

Before delving into previous literature, we introduce some terminology informally. Formal definitions, when necessary, follow in the next section. We will refer by the word *demand*, a request for sharing some d number of EPR pairs between nodes i and j. A demand is said to be *satisfied*, if the d EPR pairs are successfully shared between i and j.

Given its importance in quantum networks, the problem of establishing routes for entanglement swapping has been investigated extensively in the recent past. We now discuss briefly some of this work, for perspective.

Van Meter et al. [5] were among the earliest to study centralized quantum routing protocols. They used the shortest path first algorithm for selecting routing paths. However, they do not explicitly provision for both temporal and spatial (across multiple paths in the network) path splitting for maximizing the objective functions. Also, they did not consider time bounds between which a demand has to be satisfied. Naturally, there has been a significant body of work that employs shortest path algorithms in some form. A recent example is the work of Jian et al. [6], who use these algorithms for finding paths with fidelity guarantees using optimal entanglement purification decisions with respect to resource utilization.

Several works have leveraged the algorithms and (LP) optimization techniques available in the classical networks' domain to solve the quantum routing problem posed as flow problems. For example, Chakraborty et al. [7] posed the problem in terms of multicommodity flows. They used the LP formulation for the problem to extract the required paths, with end-to-end fidelity as a requirement.

Closer to this work, Cicconetti et al. [8] focus on request scheduling in addition to path selection. They call as scheduling, the act of allotment of a time slot in an epoch for a demand. Towards this, they develop a framework of heuristic algorithms. However, they do not focus on temporal and spatial *splitting* of the demands.

Splitting of demands across multiple paths has been noted to be superior by several works [9]. For the shared entanglement to be useful, it is important for the final state vector to have high "fidelity" – that is, be as close as possible to the maximally entangled EPR state. Several works have sought to provide guarantees on fidelity [10–12]. Opportunistic approaches for routing, both in terms of quality of next hop [10] and fastest availability [13] have been explored. Cicconetti et al. [11] also point out important differences between communication and distributed quantum communication applications, and the consequent need for different approaches for the two. Rabbie et al. [14] considered the problem of optimal placement of specialized quantum repeaters on existing classical infrastructure, satisfying requirements on EPR pair generation rates and fidelity.

[2]In case a particular swap operation fails, because for example, an EPR pair *decoheres* or a Bell State Measurement fails, then this number increases. However, this is still a good estimate.

Caleffi [15] focussed on the co-design of routing algorithms and routing metrics. He showed that a metric that is optimal for one algorithm may not be optimal for another (for example, Dijkstra or Bellman-Ford based). Chakraborty et al. [16] who studied routing in a distributed fashion, especially on the context of "virtual links" – links that are not optical fiber, but shared entanglements themselves, to be used as a resource for swapping. Indeed, there is significant ongoing research in developing quantum internet protocol stack and architecture [17]. We refer the interested reader to the excellent survey by Illiano et al. [18] for more details and further references.

2.3 Problem Statement

Input:

(1) A directed graph[3] $G = (V, E)$ representing the quantum network, where a directed edge from quantum node i to quantum node j is represented by $(i, j) \in E$. Every directed edge (i, j) is labeled by a *capacity* c_{ij}. Physically, a directed edge (i, j) with capacity $c_{ij} \in \mathbb{Z}^+$ represents the ability of node i to initiate sharing of c_{ij} EPR pairs (per unit time) with j. This graph can be encoded as an adjacency matrix A where an entry $A_{i,j}$ is an integer that captures c_{ij}.

(2) A demand matrix D, where the entry $D_{i,j}$ stands for the number of EPR pairs requested between nodes i and j, by node i.

(3) Time Constraint Matrix $T_{i,j}$, each element of which is a tuple $(s_{i,j}, e_{i,j})$, with $s_{i,j} < e_{i,j} \in \mathbb{Z}^+$. The entries $s_{i,j}$ and $e_{i,j}$ denote the lower and higher time boundaries respectively, between which the demand $D_{i,j}$ must be satisfied. The interval $[s_{i,j}, e_{i,j}]$ is a *burst*, and we refer to $e_{i,j} - s_{i,j}$, as the *burst length*. Note that $max(e_{i,j})$ is at most the epoch boundary T.

Output:

- A schedule of allocation of entangled pairs on each edge, within the time epoch T.

The output, which is essentially a schedule for the serving various demands, may optimize different objective functions under different constraints.

For example, we may seek to maximize the number of demands satisfied within the time horizon T, or the number of EPR pairs shared within T.

In this work we assume that the EPR pairs are *replenished* continuously among neighbouring vertices.

3 THE GREEDY FRAMEWORK

In this section, we discuss a greedy approach to maximize, independently, the number of demands satisfied and the number of EPR pairs distributed within the time epoch T. The same algorithmic framework works for both and differs only in the sequence in which the demands are taken up for processing. In particular, the preprocessing steps are the same.

Preprocessing Steps

(1) For each demand $D_{i,j}$, find the shortest path. The path length is defined in terms of the hops. If there are two paths of equal length, select the one with the higher value of the smallest EPR on the shortest path.

(2) As mentioned earlier, for any source and destination pair, the time taken for establishing the end-to-end entanglement is proportional to the *log* of the path length Fig. 1. Therefore, if the path length is such that the time taken for end-to-end entanglement establishment takes more than the end time constraint, we discard such demand. In other words, for a demand $D_{i,j}$, let the shortest path length be n. If $log(n) \geq e_{i,j}$ then discard the demand.

[3]We use the terms *graph* and *quantum network* interchangeably in this paper.

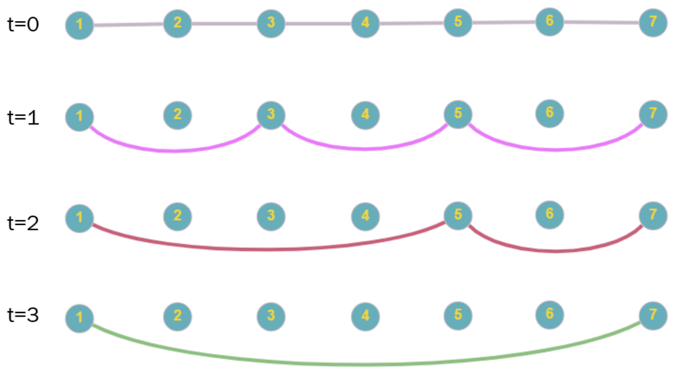


Figure 1: Illustration of time taken in establishing end-to-end entanglement.

During the pre-processing, the algorithm finds the shortest path for all the demands. For a demand $D_{i,j}$, if the length of the shortest path is such that the time for EPR sharing between i and j is more than $e_{i,j}$, then the demand is dropped in the pre-processing itself (Step 2 of pre-processing). If the shortest path itself cannot meet the deadline, longer ones will not. The $\log n$ term appears because of the nature of the entanglement swap primitive–$\Omega(\log n)$ swaps are needed for sharing entanglement between two nodes that are n hops apart.

The algorithmic framework is detailed in Algorithm 1. The sorting criterion in Step-1 is decided by the objective that we seek to maximize. We list the two sorting criteria below, before proceeding to the main algorithm.

Objective 1: Maximizing the number of demands satisfied: The demands are sorted in increasing order of $e_{i,j}$. If two demands have the same $e_{i,j}$ then priority is given to the demands with lower EPR demand. This step aims at finishing demands with fewer EPR demands first, thereby accommodating more demands on the timeline. If two demands have the same $e_{i,j}$ constraints and EPR requirements, then priority is given to the demand which has a shorter shortest path since it will take less time to establish end-to-end entanglement. If all three parameters are the same for multiple demands then the algorithm breaks the tie randomly.

Objective 2: Maximizing the number of demanded EPR pairs shared: The greedy choice for this objective function is to place demands with higher EPR requirements earlier in the sequence. The tie-breaking logic will be similar to that in the previous objective. Therefore, the algorithm will greedily pick the demands with larger EPR requirements first, thereby increasing the number of EPR pairs distributed. Note that it may impact the number of demands served and demands with lower EPR pair requirements may starve.

In the case of objectives 1 and 2, we assume that all the demands are available at t=0. In other words $\forall D_{i,j}, s_{i,j} = 0$.

Objective 3: Maximizing the number of demands satisfied with different starting times: This use case assumes that for a given time period 0-T, information about each demand is known beforehand. Additionally, the demands have different start times $s_{i,j}$. The demands must be satisfied between the lower and upper time-bound. In this use case, the greedy choice for this objective function is to arrange demands with the lower value of shortest path length followed by lower time constraint difference $(e_{i,j} - s_{i,j})$ followed by lower EPR requirement and so on. This use-case helps illustrate the concept of bursts.

We now discuss the algorithm in detail. The topology of the quantum network is defined by an adjacency matrix A. The element $(A_{i,j})$ of the adjacency matrix holds $c_{i,j}$. As the demands get allotted, the number of available EPR pairs on an edge may change. We denote by $A^t_{i,j}$, the state of the matrix at time instant t.

Because of our assumption of instantaneous replenishment, $A_{i,j}$ reduces only when a request is scheduled, reserving the shared entanglements for the scheduled demand (Steps 3 and 7 of the algorithm).

We first discuss splitting a demand $D_{i,j}$ in time, also called temporal splitting. The algorithm works by greedily choosing a demand $D_{i,j}$ from the sorted sequence of demands and schedules it fully or in part $p_1 < D_{i,j}$ at $t = 0$ on the shortest path between nodes i and j. After completion of first part p_1, the demand can be scheduled in full or in part (p_2) again, and so on until time step $e_{i,j}$ or $\sum p_i = D_{i,j}$, whichever happens earlier. Immediate scheduling along the same path is possible because of instantaneous replenishment.

In case demand is not fully met, the algorithm finds the next shortest path in the next iteration. It runs temporal scheduling again on that path until the $e_{i,j}$. This scheduling will work across multiple paths in each iteration, in increasing lengths of paths. This is the spatial scheduling of the demand across multiple paths. We only consider paths that are at most k hops in length, for a small k of our choice. The algorithm returns the count of EPR pairs, demands served and a schedule of the demands through the proposed scheduling mechanism. Fig. 2 illustrates the working of the algorithm with an example.

3.1 The GST and the Naive

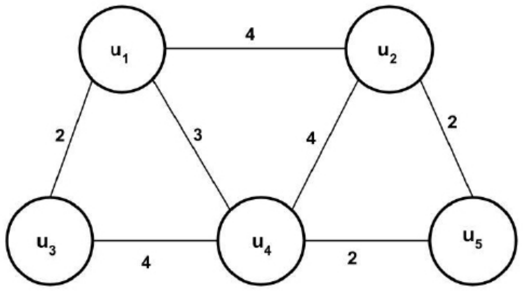
We refer to the Greedy framework above, coupled with the Spatial and Temporal splitting technique as the GST technique. To assess the quality of the solution, we use a "naive" algorithm that incorporates none of the above techniques. The naive algorithm tries to find a path for entanglement swap in the order in which the demands are presented. Further, it does not perform any temporal or spatial splitting.

3.2 Computational Complexity

We now assess the worst case time complexity of our algorithm. The worst case scenario for Algorithm 1 is a complete graph with a demand set where all demands need to be split across all the paths

Algorithm 1:

```
Initialization:
  (1) Sort the demands // Sorting criteria depends on objective
  (2) Timeline length $T \leftarrow max(e_{i,j})$
  (3) Define $A^t_{i,j} = A_{i,j}$ for $0 \le t \le T$
  (4) Number of EPR pairs distributed $E \leftarrow 0$
  (5) Number of demands served $d \leftarrow 0$
  (6) $RD$ is the ordered list of demands that are yet to be served
      or have been partially served, where each element $RD_{i,j}$
      contains the remaining EPR pairs still needed by the
      demand $D_{i,j}$ and initially $RD_{i,j} \leftarrow D_{i,j}$
  (7) Schedule variable SCHD of the size T
  (8) while $RD$ is not empty or Change in $RD_{i,j}$ over last iteration
      != 0 do
    foreach $RD_{i,j}$ do
      $itr \leftarrow 1$
      $pathLength \leftarrow$ length of the itr-th shortest path
      while $pathLength < 2^{e_{i,j}-s_{i,j}}$ do
        $t' \leftarrow s_{i,j}$
        while $t' \le e_{i,j} - log(pathLength)$ do
          let sEPR = smallest shared EPR value in $(A^{t'}_{i,j})$
          on the itr-th shortest path between $i$ & $j$
          schedule $RD_{i,j}$ for execution at $t = t'$
          if $sEPR > 0$ then
            if $sEPR < RD_{i,j}$ then
              Append [(i,j), sEPR] to SCHD[$t'$]
              $E \leftarrow E + sEPR$
              $RD_{i,j} \leftarrow RD_{i,j} - sEPR$
              Update $A^{t'}_{l,m} = A^{t'}_{l,m} - sEPR$ with $l, m$
              be the intermediate nodes on the
              shortest path for $D_{i,j}$
              $t' = t' + log(pathLength)$
            else
              Append [(i,j), $RD_{i,j}$] to SCHD[$t'$]
              $E \leftarrow E + RD_{i,j}$
              $RD_{i,j} \leftarrow 0$
              $d \leftarrow d + 1$
              Update $A^{t'}_{l,m} = A^{t'}_{l,m} - RD_{i,j}$ with $l, m$
              be the intermediate nodes on the
              shortest path for $D_{i,j}$
              $t' = e_{i,j}$
            end
          else
            $t' = t' + 1$
          end
        end
        $itr \leftarrow itr + 1$
        $pathLength \leftarrow$ length of the itr-th shortest path
      end
    end
  end
Output :E, D, SCHD
```

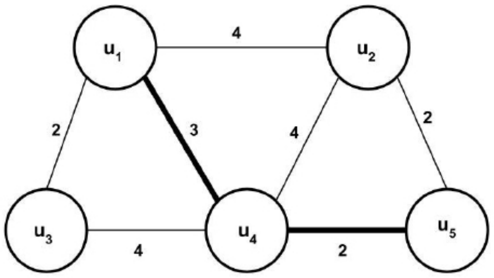


We have a graph G of 5 vertices and the capacity (number of EPR pairs that can be generated per timestep) is shown along each edge.

At timestep 0, the availability matrix looks as follows

$$A^t = \begin{bmatrix} 0 & 4 & 2 & 3 & 0 \\ 4 & 0 & 0 & 4 & 2 \\ 2 & 0 & 0 & 4 & 0 \\ 3 & 4 & 4 & 0 & 2 \\ 0 & 2 & 0 & 2 & 0 \end{bmatrix}$$

We have a remaining demand list RD = [(u1, u5, 5, 2), (u2, u4, 6, 8), (u1, u3, 5, 4)] and we illustrate the algorithm for one demand satisfaction.



The algorithm picks the first demand from this list and starts to generate as many EPR pairs as possible through the shortest path (u1-u4-u5) at t=0. Two EPR pairs can be shared through this path at timestep 1. Therefore our resource availability matrix becomes:

$$A^t = \begin{bmatrix} 0 & 4 & 2 & 1 & 0 \\ 4 & 0 & 0 & 4 & 2 \\ 2 & 0 & 0 & 4 & 0 \\ 1 & 4 & 4 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \end{bmatrix}$$

and the remaining demand list becomes

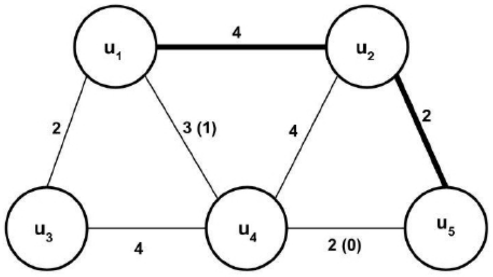
RD = [(u1, u5, 3, 2), (u2, u4, 6, 8), (u1, u3, 5, 4)]

In the next timestep, t=2, since the deadline ($t_{deadline}$ = 2) for demand has not yet passed, we make use of the same shortest path (temporal splitting) and again obtain two EPR pairs. The remaining demand list then becomes:

RD = [(u1, u5, 1, 2), (u2, u4, 6, 8), (u1, u3, 5, 4)]

The resource availability matrix A^t remains unchanged as the EPR pairs are replenished instantly.

In the next timestep t=3, since the deadline is reached, we cannot continue using the same path. However, since the demand still requires one more EPR pair, we move onto the next shortest path.



The next shortest path is (u1, u2, u5). The algorithm again begins at timestep t=0 and starts to schedule EPR generations along this path. This shows that there will be two parallel generations for the first demand at t=0; one through (u1-u4-u5) and the other through (u1-u2-u5). This is an example of spatial or path splitting. The algorithm schedules the generation of two EPR pairs through this path and the first demand will be completely satisfied.

RD = [(u2, u4, 6, 8), (u1, u3, 5, 4)]

$$A^t = \begin{bmatrix} 0 & 2 & 2 & 1 & 0 \\ 2 & 0 & 0 & 4 & 0 \\ 2 & 0 & 0 & 4 & 0 \\ 1 & 4 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 2: An example showing the working of the proposed algorithm.

at every time step. We consider at most k-hop paths for demand satisfaction in the Algorithm. The total number of k hop paths between any source-destination pair in a complete graph with $|V|$ vertices are $\sum_{j=1}^{k} \binom{|V|-2}{j-1}(j-1)!$. In the worst case, all of these paths are considered for entanglement swapping during $0 \leq t \leq T$ for all the demands $D_{i,j} \in D$. So the total time complexity of the algorithm is $O(T.|D|. \sum_{j=1}^{k} \binom{|V|-2}{j-1}(j-1)!)$, where we denote by $|D|$, the total number of demands. Notice that while having a longer epoch length T increases the time complexity, it helps in satisfying more demands. We also note that for $k = 2$, which we use in this paper, the term $\sum_{j=1}^{k} \binom{|V|-2}{j-1}(j-1)!)$ is linear in the number of quantum nodes, leading to a worst case time complexity of $O(T.|D|.|V|)$.

4 RESULTS

We now empirically show how the performance (in terms of the fraction of demands that are satisfied, and in terms of the number of EPR pairs eventually shared in the epoch $[0, T]$) scales against different size settings–larger number of nodes, larger density, more demands, more EPR requests per demand and different burst sizes. As baseline, we use a naive algorithm, that (a) does not perform spatial or temporal splitting and (b) services the demands in order of their generation.

In all the experiments, we use Erdős-Rényi (ER) random graphs. We choose ER without loss of generality–it can be a graph generated in any other manner.

Other parameters are listed below. In experiments where these parameter settings are changed, it will be stated explicitly. We emphasize that these parameters are chosen for purpose of illustration and may be changed.

- $|V|$ and p (parameters for ER graph): 50 and 0.2 respectively.
- $c_{i,j}$: Chosen uniformly at random in $[1, 5]$ for all edges.
- For a pair of nodes i, j chosen uniformly at random from a Cartesian product of nodes, we choose demands $D_{i,j}$ uniformly at random between 1 to 9.
- For a demand $D_{i,j}$, choose $e_{i,j}$ uniformly at random between 5 and 10 and $s_{i,j}$ uniformly at random between 1 and $e_{i,j}$.
- Path length limit: 2. In all the simulations, shortest paths of length 1 and 2 are considered.

We now discuss the results of these experiments in turn.

4.1 Size of the Quantum Network

To evaluate the impact of an increase in the number of quantum nodes, we change $|V|$ from 10 to 150 in steps of 10. Fig. 3 shows the results. We notice that both the fraction of demands satisfied and EPR pairs shared increases with an increase in the number of nodes–an increase in the number of nodes in an ER graph is accompanied by an increases in the number edges, and therefore paths. Therefore, the number of alternative paths available increases, which yields the result. We also note that our GST approach performs significantly better than the naive approach.

4.2 Number of demands

In this experiment, we increase the number of demands, keeping other parameters constant. The results are shown in Fig. 4. As expected, the fraction of demands satisfied and EPR pairs shared

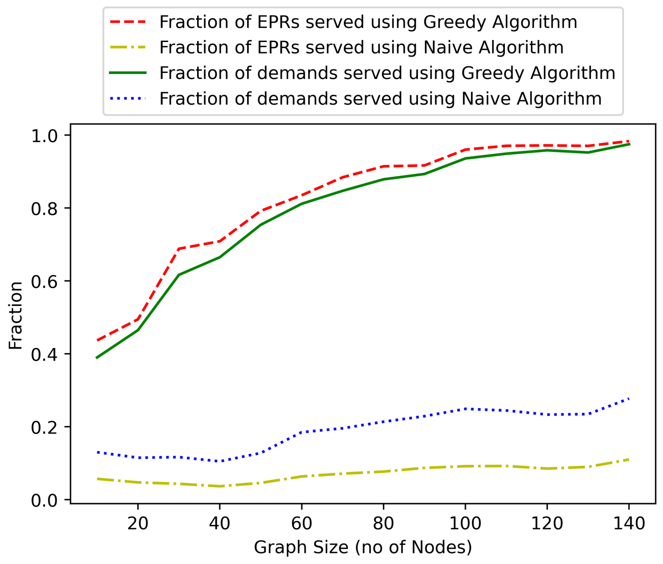


Figure 3: Plot showing the fraction of demands and EPR pairs distributed with increasing number of quantum nodes.

decreases with an increase in demand size. The naive algorithm performs significantly worse. Moreover, it deteriorates slightly with an increase in the number of demands, starting from a demand satisfaction rate of 0.16 at 100 demands to 0.14 at 1000 demands, and EPR rate of 0.052 to 0.046.

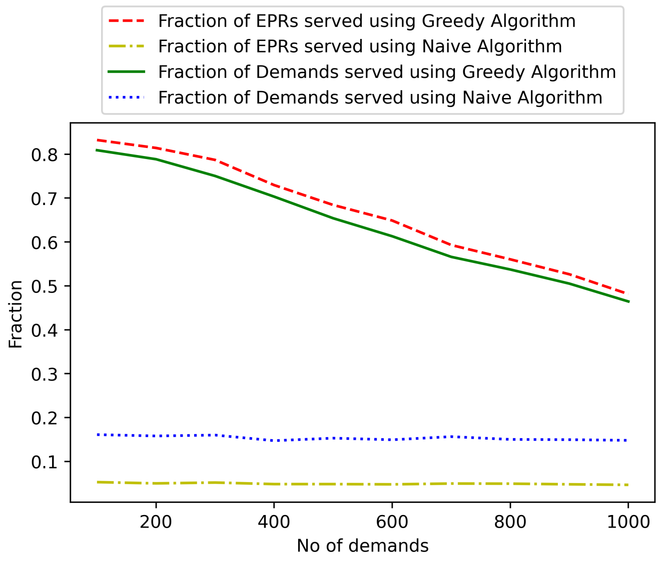


Figure 4: Plot showing the fraction of demands and EPR pairs distributed with an increasing number of demands.

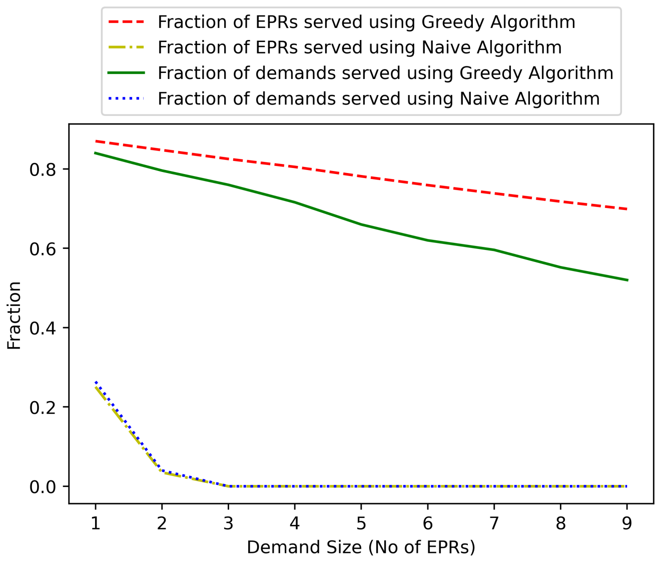


Figure 5: Plot showing the fraction of demands and EPR pairs distributed with increasing size of the EPR per demand.

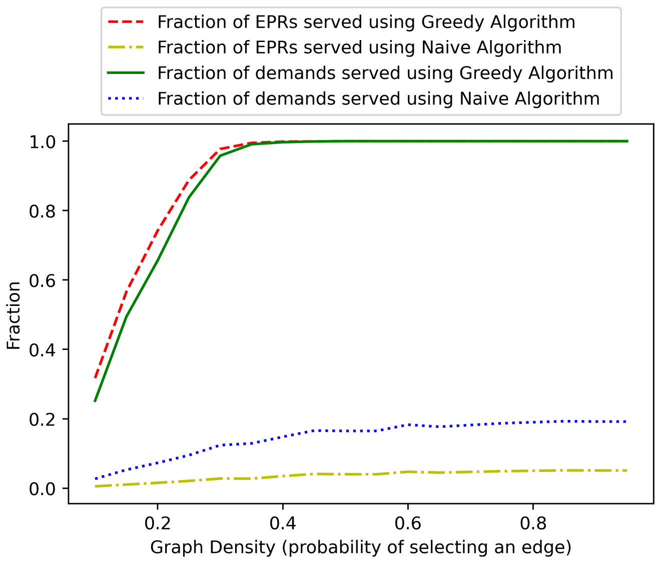


Figure 6: Plot showing the fraction of demands and EPR pairs distributed with increasing density of the graph.

4.3 Number of EPR pairs per demand

We fix the number of demands at 25. Initially the number of EPR pairs per demand is picked randomly between 1 and 3. Subsequently, we increase the number of EPR pairs per demand in steps of 1, for 10 times. With an increase in the EPR size per demand for a fixed graph, the fraction of EPR pairs shared and demands satisfied decreases almost linearly (Fig. 5). The naive algorithm fails to satisfy even one demand, after two iterations of EPR increase per demand. When the demand size increases beyond this threshold, the edge capacity $c_{i,j}$ is simply not sufficient to satisfy even one demand.

4.4 Graph density

To change the density of the graph, we changed the ER probability parameter p from 0.1 to 0.9 with an increment of 0.1. We fixed the number of demands at 100. The results are shown in Fig. 6. We see that there is a sharp rise in the fraction of demands satisfied and EPR pairs distributed, before it reaches 1. This is because an increase in density results in an easy availability of alternate paths, along which entanglement swapping can take place. After a threshold density, all demands are satisfied. The effect of providing alternative paths is so significant that even the naive algorithm performs better on denser graphs. The rate of growth in our approach is much steeper when compared to the naive algorithm because it leverages the availability of alternative paths better.

4.5 Burst Time

Bursts can have an impact on the network in multiple ways. Firstly, shorter burst requirements leave lesser room for scheduling on the network than longer burst requirements. Moreover, if many short burst demands overlap temporally, it leaves lesser room for scheduling. We report experiments to confirm this intuition.

4.5.1 Fixed Start Time with Increase in Burst Duration. In this experiment, the start time for each demand is chosen uniformly at random in the range $[0, 5]$. For each demand, the burst duration is chosen uniformly at random in $[1, 10]$. The end time of a burst is determined by adding the burst duration to the start time. The result is shown in Fig. 7. As expected, the fraction of demands satisfied and EPR pairs shared increases with an increase in the burst duration, for the GST approach. On the hand, while they increase in the naive approach as well, they hit saturation very quickly. The only increment in the fraction happens when there is flexibility in scheduling the demand in a "monolithic" manner with an increase in the burst time. However, since the naive approach does not involve temporal splitting of demands, it cannot benefit from a longer burst duration.

4.5.2 Clustered and Scattered Demands. To study the impact of clustering of demands, we design the following experiment. Clustered demands all start at 0 and end at 10. Scattered demands can have any start time in $[0, 9]$ and a burst duration of 10. Therefore, $10 \leq T \leq 20$ in this experiment. We study the effect of increasing the number of clustered and scattered demands on the network performance. The results are shown in Fig. 8. With an increasing number demands, the performance degrades in all cases. However, as expected, scattered demands result in the best performance when compared to clustered demands.

5 CONCLUSIONS AND FUTURE DIRECTIONS

In this work, we proposed a greedy framework that also employs temporal and spatial split of demands. We also introduced the idea of demands having a burst time, specified in terms of an allowable starting time and deadline to complete. We believe that this restriction is natural and relevant, considering that the lifetimes of quantum memory are limited at present. The approach discussed

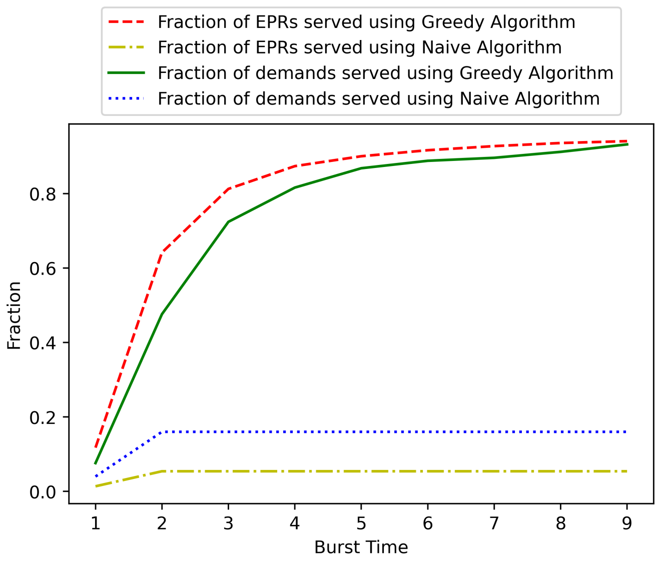


Figure 7: Plot showing the fraction of demands and EPR pairs distributed in case of increasing burst time.

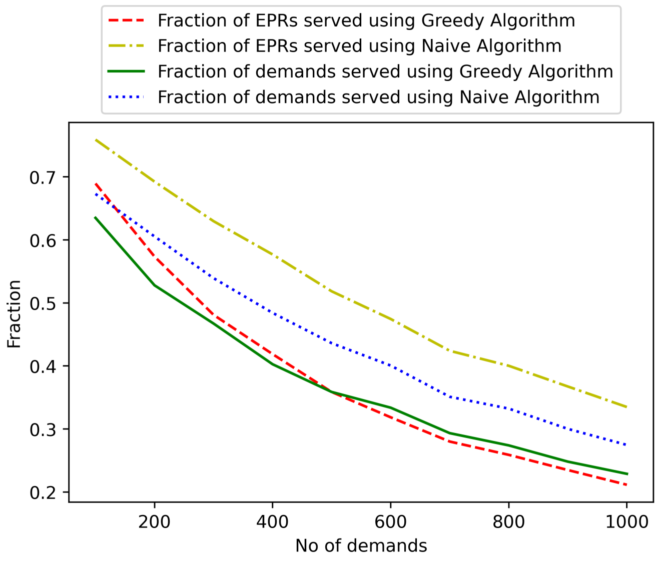


Figure 8: Plot showing the fraction of demands and EPR pairs distributed comparing clustered and scattered demands.

in this work yields an easy-to-use tool to investigate different algorithms for scheduling the demands, and the impact of burst time restrictions on network performance.

There are some aspects of routing that we ignored in this paper. How robust is the GST approach in the presence of entanglement generation and Bell State Measurement failures? How much fidelity does the approach guarantee? While investigation from these standpoints could perhaps necessitate some changes in the approach, we envisage that the required changes would be small.

This work suggests several future directions of inquiry. What are the important metrics or objective functions against which the framework can be applied? Does this greedy heuristic yield guaranteed performance bounds? Can this framework be easily modified for the "dynamic" scenario, where the demands are streaming in an online fashion?

REFERENCES

[1] Jessica Illiano, Marcello Caleffi, Antonio Manzalini, and Angela Sara Cacciapuoti. Quantum internet protocol stack: A comprehensive survey. *Computer Networks*, 213:109092, 2022.
[2] Charles H. Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K. Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Phys. Rev. Lett.*, 70:1895–1899, Mar 1993.
[3] Alexander M. Goebel, Claudia Wagenknecht, Qiang Zhang, Yu-Ao Chen, Kai Chen, Jörg Schmiedmayer, and Jian-Wei Pan. Multistage entanglement swapping. *Phys. Rev. Lett.*, 101:080403, Aug 2008.
[4] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
[5] Rodney Van Meter, Takahiko Satoh, Thaddeus D. Ladd, William J. Munro, and Kae Nemoto. Path selection for quantum repeater networks. *Networking Science*, 3(1):82–95, Dec 2013.
[6] Jian Li, Mingjun Wang, Qidong Jia, Kaiping Xue, Nenghai Yu, Qibin Sun, and Jun Lu. Fidelity-Guarantee Entanglement Routing in Quantum Networks. *arXiv e-prints*, page arXiv:2111.07764, November 2021.
[7] Kaushik Chakraborty, David Elkouss, Bruno Rijsman, and Stephanie Wehner. Entanglement distribution in a quantum network: A multicommodity flow-based approach. *IEEE Transactions on Quantum Engineering*, 1:1–21, 2020.
[8] Claudio Cicconetti, Marco Conti, and Andrea Passarella. Request scheduling in quantum networks. *IEEE Transactions on Quantum Engineering*, 2:2–17, 2021.
[9] Mihir Pant, Hari Krovi, Don Towsley, Leandros Tassiulas, Liang Jiang, Prithwish Basu, Dirk Englund, and Saikat Guha. Routing entanglement in the quantum internet. *npj Quantum Information*, 5(1):25, Mar 2019.
[10] Laszlo Gyongyosi and Sandor Imre. Opportunistic entanglement distribution for the quantum internet. *Scientific Reports*, 9(1):2219, Feb 2019.
[11] Claudio Cicconetti, Marco Conti, and Andrea Passarella. Resource allocation in quantum networks for distributed quantum computing. In *2022 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 124–132, 2022.
[12] Yuan Lee, Eric Bersin, Axel Dahlberg, Stephanie Wehner, and Dirk Englund. A quantum router architecture for high-fidelity entanglement flows in quantum networks. *npj Quantum Information*, 8(1):75, Jun 2022.
[13] Ali Farahbakhsh and Chen Feng. Opportunistic routing in quantum networks. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, page 490–499. IEEE Press, 2022.
[14] Julian Rabbie, Kaushik Chakraborty, Guus Avis, and Stephanie Wehner. Designing quantum networks using preexisting infrastructure. *npj Quantum Information*, 8(1):5, Jan 2022.
[15] Marcello Caleffi. Optimal routing for quantum networks. *IEEE Access*, 5:22299–22312, 2017.
[16] Kaushik Chakraborty, Filip Rozpedek, Axel Dahlberg, and Stephanie Wehner. Distributed routing in a quantum internet. *arXiv preprint arXiv:1907.11630*, 2019.
[17] Siddhartha Das, Sumeet Khatri, and Jonathan P. Dowling. Robust quantum network architectures and topologies for entanglement distribution. *Phys. Rev. A*, 97:012335, Jan 2018.
[18] Jessica Illiano, Marcello Caleffi, Antonio Manzalini, and Angela Sara Cacciapuoti. Quantum internet protocol stack: A comprehensive survey. *Computer Networks*, 213:109092, 2022.