A Simple Randomized Scheme for Constructing Low-Weight k-Connected Spanning Subgraphs with Applications to Distributed Algorithms

Maleq Khan * Gopal Pandurangan* V.S. Anil Kumar †

Abstract

The main focus of this paper is the analysis of a simple randomized scheme for constructing low-weight k-connected spanning subgraphs. We first show that our scheme gives a simple approximation algorithm to construct a minimum-weight k-connected spanning subgraph in a weighted complete graph, a NP-hard problem even if the weights satisfy the triangle inequality. We show that our algorithm gives an approximation ratio of $O(k \log n)$ for a metric graph, $O(k)$ for a random graph with nodes uniformly randomly distributed in $[0, 1]^2$ and $O(\log \frac{n}{k})$ for a complete graph with random edge weights $U(0, 1)$. We show that our scheme is optimal with respect to the amount of "local information" needed to compute any connected spanning subgraph. We then show that our scheme can be applied to design an efficient distributed algorithm for constructing such an approximate k-connected spanning subgraph (for any $k \geq 1$) in a point-to-point distributed model, where the processors form a complete network. Our algorithm takes $O(\log \frac{n}{k})$ time and expected $O(nk \log \frac{n}{k})$ messages. Our result in conjunction with a result of Korach et al. ([21]) implies that the expected message complexity of our algorithm is significantly better than the best distributed algorithm that finds an optimal k-connected subgraph. We also show that for the geometric instances, our randomized scheme constructs low-degree k-connected spanning subgraphs which have $O(k \log n)$ maximum degree, with high probability.

Keywords. k-Connected Spanning Subgraph, Minimum Spanning Tree, Randomized Approximation Algorithm, Distributed Algorithm, Probabilistic Analysis.

*Department of Computer Science, Purdue University, 250 N. Univ. St., West Lafayette, IN 47907, USA. E-mail: {mmkhan, gopal}@cs.purdue.edu.

†Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA. E-mail: akumar@vbi.vt.edu

1 Introduction

Computing the low weight spanning subgraphs of a given graph $G(V, E)$ with non-negative edge weights is a fundamental problem in network design (e.g., see [25, 9] for an extensive survey). One important problem in this setting is the *k-vertex connectivity problem* (henceforth simply the *k-connectivity* problem): find a spanning subgraph of minimum weight that is k-vertex-connected, i.e., there exists k vertex-disjoint paths between every pair of vertices. Finding an optimal k-connected spanning subgraph is NP-hard for $k \geq 2$ even if the weights of the edges satisfy the triangle inequality, or even when the graph is a complete Euclidean graph [14]. There has been a lot of work on designing approximation algorithms for the k-connectivity problem. Most of these algorithms are centralized algorithms which are quite sophisticated and their main goal is to obtain a polynomial time algorithms with the best possible approximation ratio (see e.g., [3, 23, 24]). Distributed algorithms for the k-connectivity ($k \geq 2$) problem has received limited attention thus far — this is especially true for the weighted version. In fact, to the best of our knowledge there is no known efficient distributed algorithm for $k \geq 2$ for weighted graphs. In contrast, for $k = 1$ — the minimum spanning tree (MST) problem — optimal distributed algorithms are well-known [7, 31]. With the emergence of the new networking technologies such as ad hoc and sensor networks, there is an increasing need for distributed algorithms that are simple and easily implementable, have low communication complexity, and perform reasonably well (e.g., see [31, 26, 32]). Such simple local algorithms are desirable even for the MST problem, where optimal distributed algorithms are known (see e.g., [11, 7, 31]), because these algorithms are quite complex, involve a lot of message complexity and synchronization to implement in a light weight and unreliable environment, such as ad hoc networks. This motivates the question of developing simple, local control, approximate algorithms. This also adds a new dimension to the design of distributed algorithms for such networks: we can potentially *tradeoff* optimality of the solution to the amount of resources (messages, time etc) consumed by the algorithm. This is the motivation for the relatively new area of *distribution approximation* (we refer to the survey by Elkin [7]).

In this paper, we study a very simple randomized scheme called *Random Nearest Neighbor (Random-NN) scheme* for constructing a low-weight k-connected spanning subgraph (for any $k \geq 1$) and show some of its properties and applications. The Random-NN scheme is based on a simple idea (cf. Section 3): each node chooses a unique *rank*, a quantity that is *randomly* chosen from a totally ordered set, and a node connects to its k *nearest* nodes of higher rank. We first show that our scheme gives a simple approximation algorithm to construct a minimum-weight k-connected spanning subgraph in a weighted complete graph, a NP-hard problem even if the weights satisfy the triangle inequality. We show that our algorithm gives an approximation ratio of $O(k \log n)$ for a metric graph, $O(k)$ for a random graph with nodes uniformly randomly distributed in $[0, 1]^2$ and $O(\log \frac{n}{k})$ for a complete graph with random edge weights $U(0, 1)$. We show that our scheme is optimal with respect to the amount of "local information" (in an average sense — defined precisely in Section 3.2) needed to compute any connected spanning subgraph.

We next show that our scheme can be applied to design an efficient distributed algorithm for constructing an approximate k-connected spanning subgraph ($k \geq 1$) in a point-to-point distributed model, where the processors form a complete network. Our algorithm takes $O(\log \frac{n}{k})$ time and expected $O(nk \log \frac{n}{k})$ messages contrasting a result of Korach et al. [21] that shows that $\Omega(n^2)$ is a lower bound of the number of messages required to find an MST (i.e., $k = 1$) in this model. Thus, the expected message complexity of our algorithm is significantly better than the best distributed algorithm that finds the (optimal) MST. The proof of this $\Omega(n^2)$ bound on the number messages for finding MST (cf. Theorem 1 in [21]) can easily be modified to show that $\Omega(n^2)$ is also a lower bound on the message complexity for finding a minimum k-connected spanning subgraph for any

$k \leq \lfloor n/2 \rfloor - 1$. This lower bound also holds for the metric weights. This also implies that our algorithm, for this restricted distributed computation model, has provably better asymptotic message complexity than the best distributed algorithm that finds a minimum k-connected subgraph, for any $k = o(n)$. However, the price for this gain is that our algorithm has a somewhat weaker approximation ratio compared to the best-known centralized algorithms.

We also show that for the geometric instances (these are relevant, for example, in the ad hoc sensor network applications [27]), our scheme constructs low-degree k-connected spanning subgraphs (these are useful in many applications e.g., see [14]) which have $O(k \log n)$ maximum degree, with high probability.

Road Map. The rest of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we present the Random-NN scheme, to construct a k-connected spanning subgraph on a given weighted complete graph. The analysis of the weight of k-connected subgraph produced by Random-NN scheme and approximation ratios for various graph models are given in Section 4. In Section 5, we describe a distributed implementation of the Random-NN scheme, and analyze its time and message complexities. We conclude in Section 6 with a discussion on future work.

2 Related Work

The work that is closest in spirit to our work is perhaps that of Imase and Waxman [18]. They consider the dynamic Steiner tree problem, where the objective is to maintain a near-optimal Steiner tree when nodes are added or deleted. They show that, under additions only (no deletions), a simple greedy algorithm which connects the just added node to nearest existing node (by the shortest path, i.e., they assume the triangle inequality) gives a $O(\log n)$ approximation. Their algorithm can be considered as a variant of our NN scheme for finding the spanning tree (i.e., the special case: $k = 1$). However, their algorithm will not work in a distributed (unlike our scheme) because we cannot connect to the shortest node (they can do that since the nodes are added one by one) as this can introduce cycles.

Random ranks were used to construct forests, by a slightly different process by Toroczkai and Bassler [36]. The process defined here chooses a random rank for each node on a graph in $G(n, p)$, and each node connects to the neighbor of highest rank. They show that the resulting forest has a power law degree distribution, which they use as a model for explaining power laws in networks.

The work of Panconesi and Rizzi [29] also uses an approach based on ranking of nodes to design simple, fast, and *deterministic* distributed algorithms to find maximal matchings, edge/vertex-colorings, and maximal independent sets. This approach however is not comparable to our Random-NN scheme because the edge weights play no role in their algorithms (they are for unweighted networks).

We now briefly mention some previous results on the centralized approximation algorithms for the k-connectivity problem ($k \geq 2$). For the general graph setting, where edge weights are arbitrary, a k-approximation algorithm is given in [23]. Cheriyan et al. [3] achieved an approximation ratio of $6H_k = O(\log k)$ for the case where $k \leq \sqrt{n/6}$. For the case where $k < (1 - \epsilon)n$, they achieved an approximation ratio of $\sqrt{n/\epsilon}$. Recently, an $O(\ln^2 k \cdot \min\{\frac{n}{n-k}, \frac{\sqrt{k}}{\ln k}\})$ approximation algorithm was given in [24].

For the metric weights (the edge weights satisfy the triangle inequality), a $2+\frac{k-1}{n}$-approximation algorithm was given in [23]. Czumaj et al. [5] presented a centralized $(1 + \epsilon)$-approximation ($\epsilon > 0$) algorithm for the minimum-weight k-connected spanning subgraph problem for a complete Euclidean graph with constant dimension. They also show that there is no polynomial time

$(1+\epsilon)$-approximation algorithm for a complete Euclidean graph in dimension $\log n$ or higher unless $P = NP$. This result also implies the same hardness of approximation in a complete metric graph.

We now mention the previous work on distributed algorithms. Most of these algorithms assume that the graph is unweighted and the goal is to find a sparse k-connected subgraph. The algorithm of Cheriyan et al. [2] finds k edge-disjoint breadth first (BFS) forests, which gives a k-connected subgraph. The distributed implementation of this algorithm has time and message complexity as $O(kn\log^3 n)$ and $O(k|E|+kn\log^3 n)$ respectively. Thurimella [35] improved the time complexity to $O(kD + kn^{0.614})$ where D is the diameter of G, but the message complexity was ignored and can be much larger than that of the algorithm given in [2]. Using similar ideas, Jennings et al. [19] developed a distributed algorithm for the k-vertex connected subgraph problem which takes $O(n)$ time and $O(|E|)$ messages. In the same paper, they also presented a distributed algorithm for the k-edge connectivity problem which takes $O((k+D)\log^3 n)$ time and $O(k|E| + kn\log^3 n)$ messages. All of these algorithms [2, 35, 19] produces a k-connected subgraph with $O(kn)$ edges from an unweighted k-connected graph G. There is also work on distributed algorithms [16, 15] for finding the biconnected components ($k = 2$, unweighted graph). Both of the algorithms given in [16] and [15] take at least linear time.

We now state relevant known distributed algorithms in the complete network model. Korach et al. [21] showed a lower bound of $\Omega(n^2)$ messages for any distributed algorithm computing a minimum weight spanning tree. This result holds even when the weights satisfy the triangle inequality. We note that our algorithm significantly beats the above lower bound at the cost of producing somewhat sub-optimal solutions. In contrast, Korach et al. gave algorithms that needed only $O(n\log n)$ messages for a class of problems that included the spanning tree problem and the leader election problem. In another paper [22], they showed that $\Omega(n\log n)$ messages are necessary for this class of problems. They also showed, however, that for the maximal matching problem and the Hamiltonian circuit problem, $\Omega(n^2)$ messages are necessary and gave the algorithms that matched this lower bound.

3 A Scheme to Construct a k-connected Subgraph in a Weighted Clique

We provide a simple scheme to construct a k-vertex connected spanning subgraph in a given complete weighted graph $\mathbb{K}_n$ of n nodes. We assume that there is a non-negative weight, $c(u,v)$ is associated with each edge (u,v) of the graph. The objective is to determine the edges that will be in the k-connected subgraph and to keep the weight of the k-connected subgraph low. The *weight of a k-connected subgraph* is the sum of the weights of the edges in it. In this section, we present a basic scheme (an abstract algorithm) and prove that this scheme indeed constructs a k-connected graph.

The scheme is quite simple. Each node u is given a unique rank $r(u)$. By *unique rank*, we mean that no two nodes have the same rank. Thus a ranking of the nodes corresponds to a permutation of the nodes. For two nodes u and v with $u \neq v$, either $r(u) < r(v)$ or $r(u) > r(v)$. Once the ranks of the nodes are chosen, they remain unchanged throughout the execution of the algorithm. Later, we will see how such ranks can be chosen. To form a k-connected graph, each node u is connected to k nearest nodes w_i, such that $r(u) < r(w_i)$ for all $1 \leq i \leq k$. Node v_1 is nearer than v_2, to u, if $c(u,v_1) < c(u,v_2)$. If $c(u,v_1) = c(u,v_2)$, break the tie arbitrarily, i.e., choose any one of v_1 and v_2 arbitrarily. The nearest nodes are chosen to minimize the weight of the constructed subgraph. However, connecting to any k higher ranked nodes produces a k-connected graph as shown below

(Proposition 3.1).

If a node does not have enough nodes of higher rank to get connected to, it is connected to the available higher ranked nodes. For example, to form a 2-connected graph, the highest ranked node does not have any such node. The second highest ranked node has only one such node, the highest ranked node. Every other node has at least two nodes to connect to. Obviously, the highest ranked node is connected to at least two other nodes, but it is not the initiator of any connection. By *"u is connected to v"*, we mean that u (the lower ranked node) is the initiator of the connection to v. We use $\eta(u)$ to denote the set of the nodes whom u is connected to.

Consider an enumeration of the nodes, $v_1, v_2, \ldots, v_n$, where v_i be the node of ith rank; for any $i > j$, $r(v_i) > r(v_j)$. In the above scheme, each node v_i is connected to the nearest $\min\{k, n-i\}$ neighbors in $\{v_{i+1}, v_{i+2}, \ldots, v_n\}$. Clearly, $|\eta(v_i)| = \min\{k, n-i\}$ and for the highest ranked node v_n, $\eta(v_n) = \phi$. We call this scheme *nearest neighbor scheme* or *NN-scheme*.

The following known proposition (Proposition 3.1) ensures that the NN-scheme constructs a k-connected subgraph.

Proposition 3.1 *Let $G = (V, E)$ be a graph on $V = \{v_1, v_2, \ldots, v_n\}$ with $n \geq k+1$ so that every v_i has at least $\min\{k, n-i\}$ neighbors in $\{v_{i+1}, v_{i+2}, \ldots, v_n\}$. Then G is k-connected.*

Proof: If $n = k+1$, then G is a complete graph. Assume that $n \geq k+2$ and suppose to the contrary that G is not k-connected. Then there is a $C \subseteq V$ with $|C| \leq k-1$ so that $G - C$ is disconnected. Let X, Y be two distinct connected components of $G - C$, and let $x = \max_{v_i \in X} i$ and $y = \max_{v_i \in Y} i$. For any $i > x$, if v_i is a neighbor of v_x, then v_i must be in C. Now v_x has at most $k-1$ (since $|C| \leq k-1$) and at least $\min\{k, n-x\}$ neighbors in $\{v_{x+1}, v_{x+2}, \ldots, v_n\}$. Thus, we must have $\{v_{x+1}, v_{x+2}, \ldots, v_n\} \subseteq C$; hence $x > y$. The same argument applied on v_y gives $y > x$. Thus we have a contradiction. □

Nearest Neighbor Tree (NNT). When $k = 1$, the NN-scheme produces a spanning tree. If $k = 1$, each node (except the highest ranked node) connects to exactly one higher ranked one. Thus there are $n-1$ edges in the resulting graph. If we consider each edge is directed from the lower ranked node to the higher ranked node, it is easy to see that there is no cycle in this graph. Therefore, the resulting graph is a tree spanning all n nodes. We call this spanning tree a *nearest neighbor tree*, or in short, NNT.

We use the following definitions and notations in the rest of the paper.

Let $\mathbb{N}_u(i)$ denote the i^{th} nearest neighbor of u in the given complete graph $\mathbb{K}_n$.

Definition 3.1 ***i-neighborhood.*** *The i-neighborhood of a node u, denoted by $\Gamma_u(i)$, is the set of the i nearest neighbors of u in $\mathbb{K}_n$; i.e., $\Gamma_u(i) = \{\mathbb{N}_u(1), \mathbb{N}_u(2), \ldots, \mathbb{N}_u(i)\}$. Define $\Gamma_u(0) = \phi$.*

Definition 3.2 ***jth connection.*** *Let $w_1, w_2, \ldots, w_{|\eta(u)|}$, in non-decreasing order of $c(u, w_t)$, be the nodes in $\eta(u)$. The connection u makes to w_j, for any $1 \leq j \leq |\eta(u)|$, i.e., the edge (u, w_j), is called the jth connection of u.*

3.1 Random Ranking

While ranks can be chosen in many ways, in this paper, we focus on a simple randomized way of choosing ranks: each node chooses a rank uniformly and independently at random from a totally ordered set. A random ranking can be chosen as follows. We assume that each node u has a unique identifier, $id(u)$. Generate a random number $p(u) \in [0, 1]$ for each node u. Now define, for any two node u and v, $r(u) < r(v)$ iff $p(u) < p(v)$ or $p(u) = p(v)$ and $id(u) < id(v)$. Note that the

identifiers of the nodes also constitute a ranking of the nodes. However, here we are interested in a random ranking. We will see later, using random ranking, in contrast to an arbitrary ranking, we can have a better bound on the weight of the k-connected subgraph given by the NN-scheme and on the time and message complexity of the distributed implementation of the NN-scheme. Henceforth, we call the NN scheme with the random ranking as the *Random-NN scheme*.

Later, in the analysis of weight, time, and message complexity, we will use the following lemma regarding the random ranking of the nodes.

Lemma 3.1 *When a random ranking is used, the probability that an arbitrary node u makes the jth connection to $\mathbb{N}_u(i)$ is $\frac{j}{i(i+1)}$ for $i \geq j$.*

Proof: Node u makes the jth connection to $\mathbb{N}_u(i)$ if and only if $r(\mathbb{N}_u(i)) > r(u)$ and there are exactly $j-1$ nodes in $\Gamma_u(i-1)$ with ranks higher than $r(u)$. That is, $r(u)$ is exactly $(j+1)$st among the ranks of these $i+1$ nodes (u and the i nodes in $\Gamma_u(i)$) and $\mathbb{N}_u(i)$ is one of the j highest ranked nodes among the i nodes in $\Gamma_u(i)$.

Thus, the desired probability is $\frac{1}{i+1} \times \frac{j}{i} = \frac{j}{i(i+1)}$ for $i \geq j$. $\square$

Remarks: 1) It is not possible for u to make the jth connection to a node closer than $\mathbb{N}_u(j)$.
2) The probability that u is able to make the jth connection is $\sum_{i=j}^{n-1} \frac{j}{i(i+1)} = 1 - \frac{j}{n}$. That is, j out of n nodes do not have their jth connection.

3.2 Average Neighborhood Size in Random-NN Scheme

In the NN-scheme, a node has to find the k closest nodes of higher rank to connect to. For a node u, let $v_1, v_2, \ldots v_i, \ldots$ be the nodes, in non-decreasing order of $c(u, v_i)$, i.e., v_i is the ith nearest neighbor of u. For a given choice of ranks, let $s(u)$ be the number of nodes that u has to examine (starting from v_1) before it finds the required number of nodes of higher rank. We call $s(u)$ the *size of the neighborhood*, which u has to look for, in order to find the connecting edges. The size of the neighborhood measures the amount of *local information* needed by a distributed algorithm. The quantity $s(u)$ has a bearing on the message complexity in distributed implementation (Section 5). For arbitrary choices of ranks, the average neighborhood size (i.e., $(1/n)\sum_u s(u)$) could be $\Omega(n)$. The following lemma shows that the average neighborhood size decreases significantly if we use the random ranking (Random-NN scheme). The notation H_n is used to denote the harmonic series $\sum_{i=1}^{n} \frac{1}{i} = \Theta(\log n)$.

Lemma 3.2 *Let an arbitrary node u makes the k-th connection to $\mathbb{N}_u(L)$. Then $E[L] = k(H_n - H_k) = \Theta(k \log \frac{n}{k})$.*

Proof: Using Lemma 3.1,

$$E[L] = \sum_{i=k}^{n-1} \frac{k}{i(i+1)} i = k(H_n - H_k).$$

$\square$

The above result shows that an efficient distributed algorithm can potentially be developed for the Random-NN scheme. Consider an algorithm where each node examines its neighbors beginning from the nearest neighbor until it finds the connecting edges. Lemma 3.2 says that using a random ranking, on average, each node needs information from $\Theta(k \log \frac{n}{k})$ nearest neighbors. This is optimal in general, because this is the optimal local information needed to find any spanning tree ($k = 1$) on a complete network. Korach et al. [21, 22] showed that any distributed algorithm

that constructs a spanning tree in a complete graph uses $\Omega(n \log n)$ edges. That is, on average, each node needs to use $\Omega(\log n)$ edges; i.e., each nodes needs information from at least $\Omega(\log n)$ other nodes. Thus average neighborhood size for any spanning tree is at least $\Omega(\log n)$. As a result, in terms of locality, Random-NN scheme can be said to be optimal in general.

Another result by Korach et al. [21] implies that a much larger locality is required to find a minimum spanning tree (MST). They showed that any distributed algorithm to find an MST on a complete weighted graph uses $\Omega(n^2)$ edges. The proof of this $\Omega(n^2)$ bound on the number of messages for finding an MST (cf. Theorem 1 in [21]) can easily be modified to show that $\Omega(n^2)$ is also a lower bound on the number of messages for finding an optimal k-connected spanning subgraph for any $k \leq \lfloor n/2 \rfloor - 1$. This lower bound can be shown to hold also for a complete metric graph. That is, each node uses information from $\Omega(n)$ other nodes on the average. Thus, the average neighborhood size to find an optimal k-connected subgraph is $\Omega(n)$, which is exponentially larger than that needed by the Random-NN scheme.

4 Weight of the k-Connected Subgraph

We analyze the weight of the k-connected graph constructed by the NN scheme with respect to the minimum weight k-connected (sub)graph. Throughout the rest of the paper, we use G_k and MKG to denote the k-connected graph constructed by the NN scheme and a minimum weight k-connected graph, respectively.

Let $G = (V, E, W)$ be any weighted undirected graph, where V is the set of vertices, E is the set of edges and $W =< c(u, v) >$, where $c(u, v) \geq 0$ is the weight of the edge $(u, v) \in E$. The weight of G is defined by $c(G) = \sum_{(u,v)\in E} c(u, v)$.

Using the following known proposition (Proposition 4.1), we have $c(MKG) \geq \frac{k}{2} c(MST)$. Later, we use this lower bound of $c(MKG)$ to obtain an upper bound for the approximation ratio $c(G_k)/c(MKG)$.

Proposition 4.1 *Any k-edge-connected graph G has a spanning tree T with $c(T) \leq 2c(G)/k$.*

Proof: Let D be the bidirection of G; i.e., for each edge (u, v) in the undirected graph G, there are two directed edges (u, v) and (v, u) in the directed graph D. Let w be any node in G. In the graph G, there are k edge-disjoint paths from w to any other node. Then, in D, there are k edge-disjoint directed paths from w to any other node. Edmonds [6] proved that if a directed graph has k edge disjoint paths from a node w to any other node, then it contains k edge-disjoint arborescences rooted at w. Thus D contains k edge-disjoint arborescences rooted at w. Let T be the underlying tree of the least weight arborescence among them. Then $c(T) \leq c(D)/k = 2c(G)/k$. □

We can find an example where $c(MKG)$ is exactly equal to $\frac{k}{2} c(MST)$. This shows that this lower bound for the weight of MKG is tight. It is possible to construct a k-connected graph having exactly $\frac{kn}{2}$ edges. Consider a k-cube graph where weight of each edge is one unit. Number of nodes in a k-cube graph is $n = 2^k$. Each node is uniquely identified by a k-tuple $< b_1, b_2, \ldots, b_k >$ where $b_i \in \{0, 1\}$ for $1 \leq i \leq k$. There is an edge between any two nodes u and v if and only if the k-tuples of u and v differ in exactly one component. A k-cube graph is k-connected and the degree of each node is k. Thus, the number of edges is $\frac{kn}{2}$. The weight of this k-connected graph is $\frac{kn}{2}$ and the weight of an MST on this graph is $n-1$. The ratio of these weights is $\frac{kn}{2(n-1)}$, which approaches $\frac{k}{2}$ as $n \to \infty$.

Next we analyze the weight of G_k (output of the NN scheme) and its approximation ratios to MKG for graphs with edge weights satisfying various characteristics.

4.1 Metric Graph

A metric graph is a complete weighted graph where the weights of the edges satisfy the triangle inequality. We show that for a metric graph, using any arbitrary ranking of the nodes, the NN scheme outputs a k-connected subgraph with approximation ratio of $O(k \log n)$ to MKG (Theorem 4.1).

In the rest of this section, we use I_k to denote the sum of the first k positive integers, i.e., $\sum_{i=1}^{k} i = \frac{1}{2}k(k+1)$.

Theorem 4.1 *On a metric graph G of n nodes, for any arbitrary ranking of the nodes, the weight of the k-connected graph G_k constructed by the NN-scheme, $c(G_k) = O(k \lg n)c(MKG)$, where MKG is a minimum k-connected subgraph of G.*

Proof: The theorem holds trivially for $n \leq k$. The following proof is constructed for $n \geq k+1$.

Construct a hamiltonian path S such that $c(S) \leq 2c(MST)$, where MST is a minimum spanning tree on G. Such a path S can be constructed as follows (e.g., see [4]): select any node to be the root of the MST and perform a preorder tree walk on the MST. Let the order of the nodes, as they are visited in the preorder walk, be $v_1, v_2, \ldots, v_n$. (Note that this order of the nodes is used only to construct S. To construct G_k, we assume an arbitrary ranking, which can be different from this ordering, of the nodes.) Now, add the edges (v_i, v_{i+1}) to S, for $i = 1, 2, \ldots, n-1$.

For any i, j such that $1 \leq i \leq j \leq n$, let $S_{i,j}$ denotes the sub-path $< v_i, v_{i+1}, \ldots, v_j >$ and $V_{i,j}$ denotes the subset $\{v_i, v_{i+1}, \ldots, v_j\}$. Let $G_{i,j}$ be the subgraph of G induced by $V_{i,j}$, and $F_{i,j}$ be the k-connected subgraph produced by the NN scheme running on $G_{i,j}$. Now, by induction on the number of nodes $|V_{i,j}|$, we show that for any i and j such that $|V_{i,j}| \geq k+1$,

$$c(F_{i,j}) \leq 2I_k c(S_{i,j}) \lg |V_{i,j}|. \tag{1}$$

The basis of the induction is any i, j such that $k+1 \leq |V_{i,j}| \leq 2k+1$. The number of edges in $F_{i,j}$ is $k|V_{i,j}| - I_k$. Since the weights of the edges satisfy the triangle inequality, the weight of any edge in $F_{i,j}$ is at most $c(S_{i,j})$. Thus, we have

$$c(F_{i,j}) \leq (k|V_{i,j}| - I_k)c(S_{i,j}) \leq (k(2k+1) - I_k)c(S_{i,j}) \leq 2I_k c(S_{i,j}) \lg |V_{i,j}|$$

by assuming $|V_{i,j}| \geq 3$. For $|Vi, j| = 2$, Inequality 1 holds trivially for any $k \geq 1$.

Now we show the induction step. Consider any i, j such that $|V_{i,j}| \geq 2k+2$. Let $m = |V_{i,j}|$ and $x = \lfloor (i+j)/2 \rfloor$. By the induction hypothesis,

$$\begin{aligned} c(F_{i,x}) &\leq 2I_k c(S_{i,x}) \lg |V_{i,x}| = 2I_k c(S_{i,x}) \lg \lceil m/2 \rceil, \\ c(F_{x+1,j}) &\leq 2I_k c(S_{x+1,j}) \lg |V_{x+1,j}| = 2I_k c(S_{x+1,j}) \lg \lfloor m/2 \rfloor. \end{aligned}$$

For any node $v \in V_{i,x}$, if w_1, w_2 are the t^{th} closest (to v) nodes of higher rank in $V_{i,x}$ and $V_{i,j}$, respectively, then $c(v, w_2) \leq c(v, w_1)$; a similar statement holds for any node in $V_{x+1,j}$. Therefore, for any node v, the weight of the t^{th} connection chosen by v in $F_{i,x}$ or $F_{x+1,j}$ is at least as much as that in $F_{i,j}$. Graph $F_{i,j}$ has I_k more edges than the combined edges of $F_{i,x}$ and $F_{x+1,j}$. The weight of each such edge is at most $c(S[i,j])$. Therefore,

$$\begin{aligned} c(F_{i,j}) &\leq c(F_{i,x}) + c(F_{x+1,j}) + I_k c(S_{i,j}) \\ &\leq 2I_k c(S_{i,x}) \lg \lceil m/2 \rceil + 2I_k c(S_{x+1,j}) \lg \lfloor m/2 \rfloor + I_k c(S_{i,j}) \\ &\leq 2I_k \{c(S_{i,x}) + c(S_{x+1,j})\} \lg \lceil m/2 \rceil + I_k c(S_{i,j}) \\ &\leq 2I_k c(S_{i,j}) \lg \lceil m/2 \rceil + I_k c(S_{i,j}) \\ &\leq 2I_k c(S_{i,j}) \lg |V_{i,j}|, \end{aligned}$$

where the last inequality holds for $|V[i,j]| \geq 3$. Therefore, by construction of S,

$$c(G_k) = c(F_{1,n}) \leq 2I_k c(S) \lg n \leq 4I_k c(MST) \lg n. \quad (2)$$

The weight of the optimal k-connected graph $c(MKG) \geq \frac{k}{2} c(MST)$. Thus, we have

$$c(G_k) \leq 4(k+1)(\lg n) c(MKG).$$

□

Remarks. 1. Putting $k = 1$ in Inequality 2, we get $c(NNT) = c(G_1) \leq 4(\log n)c(MST)$. However, for this special case, $k = 1$, with the help of a lemma by Rosenkrantz, Stearns, and Lewis [33, Lemma 1] concerning the traveling salesman problem, we can achieve a better bound of $\lceil \log n \rceil c(MST)$, improved by a factor of 4.
2. The above bound is asymptotically tight in general. Consider a geometric instance where n nodes are placed on a straight line equally apart by a unit distance and the weight of the between any two nodes is their distance on the line. There is a ranking of the nodes, for which, the weight of the NNT (i.e., $k = 1$) is $\Theta(n \log n)$. In fact, a random ranking of nodes (i.e., the Random-NN scheme) can be shown to give a spanning tree of the expected weight $\Theta(n \log n)$. The weight of MST on this geometric instance is $\Theta(n)$, which gives an approximation factor of $\Theta(\log n)$.

Notice that the above theorem also applies to an important special case, namely that of a geometric graph: the nodes are coordinates in a d-dimensional space and the weight of the edge between any two nodes is the Euclidean distance (or any Minkowski distance) between them. In the next section, using the Euclidean distance, we show that the algorithm yields a better approximation of $O(k)$ when nodes are randomly distributed in a 2-dimensional space.

4.2 Random Graph with Uniform Distribution of Nodes on a Plane

In this section, we analyze the weight of the k-connected graph given by the Random NN-scheme in a complete geometric graph where n nodes are randomly and uniformly distributed in a unit square $[0,1]^2$ and the weight of the edge between any two nodes is the Euclidean distance between them. In this model, the probability that a particular node lies within a particular region inside the unit square is directly proportional to the area of the region. We show the following theorem:

Theorem 4.2 *For n points distributed randomly and uniformly in $[0,1]^2$, the approximation guarantee of the Random-NN scheme is $E[c(G_k)]/E[c(MKG)] = O(k)$.*

To show the above theorem we first upper bound the weight of the k-connected subgraph constructed by the Random-NN scheme.

Lemma 4.1 *For n points distributed randomly and uniformly in $[0,1]^2$, the expected weight of G_k, the subgraph constructed by the Random NN-scheme, is $O(k^2\sqrt{n})$, i.e., $E[c(G_k)] = O(k^2\sqrt{n})$.*

Proof: Consider an arbitrary node u, and the concentric circles centered at u with radii $r_i = \frac{2^i}{\sqrt{n}}$ for $i = 1, 2, \ldots, m$. Considering a unit square, the maximum distance between any two nodes is $\sqrt{2}$. Thus, $r_{m-1} < \sqrt{2} \leq r_m$, i.e., the maximum number of these circles is $m < \frac{1}{2}\lg n + \frac{3}{2}$. Let C_i be the set of the nodes in the circle with the radius r_i and $R_i = C_i - C_{i-1}$ for $i \geq 2$ and $R_i = C_i$ for $i = 1$. For a node $v \in R_i$, The weight of the edge (u,v) is $c(u,v) \leq r_i$.

Let A_i be the event that u makes the jth connection to a node $v \in R_i$. By Lemma 3.1, the probability that u makes the jth connection to any node in $\Gamma_u(y-1) - \Gamma_u(x-1)$ is $\sum_{i=x}^{y-1} \frac{j}{i(i+1)} = \frac{j}{x} - \frac{j}{y}$,

where $j \leq x < y$. For $i \geq 2$, $|C_{i-1}| \geq 1$ since C_{i-1} contains at least one node, which is u. Considering the fact that u can be close to the border of the unit square, the probability that a particular node, other than u, is in C_{i-1} is $p \geq \frac{1}{4}$ of the area of $C_{i-1} = \frac{1}{4}\pi r_{i-1}^2 = \frac{2^{2i}\pi}{16n}$. Thus for $i \geq 2$,

$$\begin{aligned}
\Pr\{A_i\} &= \sum_{x=j}^{n}\sum_{y=x}^{n}\left(\frac{j}{x}-\frac{j}{y}\right)\Pr\{|C_{i-1}| = x \wedge |C_i| = y\} \\
&\leq \sum_{x=1}^{n}\sum_{y=x}^{n}\frac{j}{x}\Pr\{|C_{i-1}| = x \wedge |C_i| = y\} = \sum_{x=1}^{n}\frac{j}{x}\Pr\{|C_{i-1}| = x\} \\
&= \sum_{x=1}^{n}\frac{j}{x}\binom{n-1}{x-1}p^{x-1}(1-p)^{n-x} = \frac{j}{np}\left\{1-(1-p)^n\right\} \leq \frac{j}{np} \leq \frac{16j}{2^{2i}\pi}.
\end{aligned}$$

Let $c_j(u)$ be the weight of the edge given by the jth connection of u. We get

$$\begin{aligned}
E[c_j(u)] &\leq \Pr\{A_1\}r_1 + \sum_{i=2}^{m}\Pr\{A_i\}r_i \\
&\leq r_1 + \sum_{i=2}^{m}\frac{16j}{2^{2i}\pi}r_i = \frac{1}{\sqrt{n}}\left(2+\frac{8j}{\pi}-\frac{4\sqrt{2}j}{\pi\sqrt{n}}\right)
\end{aligned}$$

By linearity of expectation for all connections of n nodes,

$$E[c(G_k)] = n \times \sum_{j=1}^{k}E[c_j(u)] \leq \sqrt{n}\left\{2+\frac{8I_k}{\pi}\right\} - \frac{4\sqrt{2}I_k}{\pi} = O(k^2\sqrt{n}).$$

□

Proof: (of Theorem 4.2) It is well-known that the weight of an MST in the above graph model is $\Theta(\sqrt{n})$ (e.g., [34]). The weight of the optimal k-connected graph $c(MKG) \geq \frac{k}{2}c(MST) = \Theta(k\sqrt{n})$. Thus from Lemma 4.1, we have an approximation ratio of $O(k)$. □

4.3 Graph with Random Edge Weights

In this section, we analyze the weight of the k-connected subgraphs in another well-studied random graph model (e.g., see [10, 8, 12]) where the weights of the edges are selected randomly from $[0, 1]$ according to a uniform distribution, i.e., $U(0, 1)$. The following theorem shows the approximation guarantee of Random-NN scheme.

Theorem 4.3 *The approximation guarantee of the Random NN-scheme on a complete graph $\mathbb{K}_n$, where the weights of the edges are chosen randomly following the distribution $U(0, 1)$ is $2H_n - 2H_{k+1} + 1 = O(\log \frac{n}{k})$.*

We note that this model does not necessarily generate a metric graph, but our algorithm still gives a significantly better approximation of $O(\log \frac{n}{k})$. Frieze [10] showed that in this model, the expected weight of the MST converges to a constant $\zeta(3) = 1.202\cdots$ as $n \to \infty$. Here we show a lower bound of $\frac{1}{2}I_k$ for the expected weight of the MKG (Lemma 4.3) and show that the expected weight of G_k is $I_k(H_n - H_{k+1} + \frac{1}{2})$ (Lemma 4.4). Thus, we have an approximation ratio of $2H_n - 2H_{k+1} + 1 = O(\log \frac{n}{k})$. We now proceed to show the following lemmas, which prove the above theorem.

The proof of Lemma 4.2 can be found in [28, Page 195].

Lemma 4.2 *[28] Let X_i be the ith smallest number among n independent uniform random variables over $[0,1]$. Then $E[X_i] = \frac{i}{n+1}$.*

Lemma 4.3 *Let MKG be a minimum weight k-connected subgraph on a complete graph $\mathbb{K}_n$, where the weights of the edges are randomly chosen according to the uniform distribution $U(0,1)$. Then $E[c(MKG)] \geq \frac{1}{2} I_k$.*

Proof: Consider an arbitrary node u. Let the weights of the $n-1$ edges adjacent to u in $\mathbb{K}_n$ be $e_1, e_2, \ldots, e_{n-1}$ in non-decreasing order. These edge weights are chosen randomly and independently from $U(0,1)$. Thus, by Lemma 4.2, $E[e_i] = \frac{i}{n}$. Since the MKG is k-connected, the degree of each node in the MKG is at least k. Thus the sum of the weights of the edges adjacent to u in MKG is at least $\sum_{i=1}^{k} e_i$ and the expected sum of the weights is at least

$$E\left[\sum_{i=1}^{k} e_i\right] = \sum_{i=1}^{k} E[e_i] = \frac{1}{n} I_k$$

Using the fact that each edge is counted by at most two nodes and by linearity of expectation for n nodes,

$$E[c(MKG)] \geq \frac{1}{2} \times n \times \frac{1}{n} I_k = \frac{1}{2} I_k$$

□

Lemma 4.4 *Let G_k be the k-connected subgraph given by the Random-NN scheme on a complete graph $\mathbb{K}_n$, where the weights of the edges are chosen randomly according to the distribution $U(0,1)$. Then $E[c(G_k)] = I_k(H_n - H_{k+1} + \frac{1}{2})$.*

Proof: Again, consider an arbitrary node u. Let the weight of the $(n-1)$ edges adjacent to u in $\mathbb{K}_n$ be $e_1, e_2, \ldots, e_{n-1}$ in non-decreasing order. Then $E[c(u, \mathbb{N}_u(i))] = E[e_i] = \frac{i}{n}$ (Lemma 4.2).

The event that u makes the jth connection to $\mathbb{N}_u(i)$, $j \leq i$, is independent of the weights of the edges adjacent to u. By using Lemma 3.1, the expected weight of the jth connection by u is

$$\sum_{i=j}^{n-1} \frac{j}{i(i+1)} E[e_i] = \frac{j}{n}(H_n - H_j)$$

Using linearity of expectation, the expected total weight of all connections by the n nodes is

$$E[c(G_k)] = n \sum_{j=1}^{k} \frac{j}{n}(H_n - H_j) = I_k H_n - \sum_{j=1}^{k} j H_j$$

Using the identity $\sum_{j=1}^{k} j H_j = I_k(H_{k+1} - 1/2)$ (see [13], Page 56, Eq. 2.57),

$$E[c(G_k)] = I_k(H_n - H_{k+1} + 1/2)$$

□

4.4 Maximum Degree in the Geometric Instances

We assume that the nodes are points in a d-dimensional space and the weight of an edge between any two nodes is the Euclidean distance between them. We show the following theorem:

Theorem 4.4 *In a geometric graph, the maximum degree of a node in the k-connected spanning subgraph constructed by the Random-NN scheme is $O(k \log n)$ with high probability, i.e., with probability at least $1 - 1/n^{\Omega(1)}$.*

We show the result assuming $d = 2$, i.e., the nodes (points) are on a plane; however, this result can be generalized to any constant d. Note that for analyzing the maximum degree of a node, we do not assume any particular distribution of the nodes; we consider an arbitrary placement of the nodes in a plane. To show the desired bound on the maximum degree, we first need the following lemma.

Lemma 4.5 *Let V be the set of the nodes in the plane. If a node $v \in V$ makes its longest connection, i.e., the $|\eta(v)|^{th}$ connection, to node w, we say that a charge of 1 is placed on every node u in the closed ball $\mathcal{B}(v, c(v, w))$, where $c(u, w)$ is the weight of the edge (u, w), i.e., the distance between u and w. Then, the total charge on any node u is $O(k \log n)$, with high probability.*

Proof: Consider any node u, and partition the 2π angle around u into 6 cones with each of the angles be $\pi/3$. Consider one such cone. We prove that the total charge on u from the nodes in this cone is $O(k \log n)$, with high probability. Order the points in the cone as $v_1, v_2, v_3, \ldots$ in non-decreasing order of their distances from u (see Fig. 1). Node v_i places a charge on u *only if* the rank of v_i is in the top $|\eta(v_i)|$ among the ranks of the nodes $v_1, v_2, \ldots v_i$. Thus, the probability that v_i places a charge on u is at most $|\eta(v_i)|/i \leq k/i$. Therefore, the total expected charge on u from these nodes is at most $\sum_{i=1}^{n-1}(k/i) \leq k \log n$.

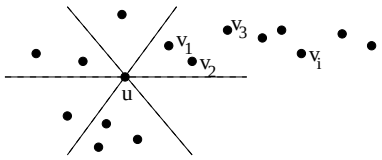

Figure 1: Each wedge around the node u is $60°$. $v_1, v_2, v_3 \ldots$ are the nodes in one wedge in non-decreasing order of their distances from u.

In order to bound the maximum charge on any node, we use a variant of the Chernoff bound [Lemma 4.6] that holds in the presence of dependencies among the variables.

Lemma 4.6 *([30]) Let $X_1, X_2, \ldots, X_l \in \{0, 1\}$ be random variables such that for all i, and for any $S \subseteq \{X_1, \ldots, X_i\}$, $\Pr[X_{i+1} = 1 | \bigwedge_{j \in S} X_j = 1] \leq \Pr[X_{i+1} = 1]$. Then for any $\delta > 0$, $\Pr[\sum_i X_i \geq \mu(1 + \delta)] \leq (\frac{e^\delta}{(1+\delta)^{1+\delta}})^\mu$, where $\mu = \sum_i E[X_i]$.*

Let $\mathcal{E}(v)$ be the event that v places a charge on u. In order to use the Chernoff bound, we need to show that, for any i, and any subset $S \subset \{v_1, \ldots, v_i\}$, $\Pr[\mathcal{E}(v_{i+1}) | \bigwedge_{w \in S} \mathcal{E}(w)] \leq \Pr[\mathcal{E}(v_{i+1})]$.

First, suppose $c(w, v_{i+1}) \geq c(w, u)$ for each $w \in S$. Then, the events $\bigwedge_{w \in S} \mathcal{E}(w)$ do not place any constraint on $rank(v_{i+1})$, relative to $rank(v_j), j \leq i$, and therefore, $\Pr[\mathcal{E}(v_{i+1}) | \bigwedge_{w \in S} \mathcal{E}(w)] = \Pr[\mathcal{E}(v_{i+1})]$.

Next, suppose $c(w, v_{i+1}) < c(w, u)$ for some $w \in S$. If the event $\mathcal{E}(w)$ occurs, then $rank(w)$ is in the top $|\eta(w)|$ ranks among the ranks of the nodes $v_1, v_2, \ldots v_{i+1}$, and the probability of $rank(v_{i+1})$ being in the top $|\eta(v_{i+1})|$ ranks goes down; that is, $\Pr[\mathcal{E}(v_{i+1}) | \bigwedge_{w \in S} \mathcal{E}(w)] \leq \Pr[\mathcal{E}(v_{i+1})]$.

Next, we apply the Chernoff bound with $\delta = \frac{5k \log n}{\mu} - 1$, where μ is the expected charge on u. Since $\mu \leq k \log n$, $\delta > 0$. Let X be the total charge on u. Then,

$$\Pr\{X \geq 5k \log n\} = \Pr\{X \geq (1+\delta)\mu\} < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \leq \left(\frac{e}{1+\delta}\right)^{(1+\delta)\mu} \leq \frac{1}{n^{3k}}.$$

Thus, with probability at least $1 - 1/n^{3k}$, where $k \geq 1$, the total charge on u is $O(k \log n)$. Using the union bound, this holds simultaneously for all nodes with probability at least $1 - 1/n^{2k}$. □

Proof: (of Theorem 4.4) If a node u connects to v, u must place a charge on v (see Lemma 4.5). Thus, the total charge on v is an upper bound on the number of nodes that are connected to v. Further, $\eta(v) \leq k$. Thus, the degree of v is at most $k + O(k \log n) = O(k \log n)$ with probability at least $1 - 1/n^{2k}$. □

5 Distributed Implementation

In this section, we give an efficient distributed implementation of the Random-NN scheme. Our distributed algorithm takes $O(\log \frac{n}{k})$ time and expected $O(nk \log \frac{n}{k})$ messages to construct a k-connected graph.

Model of distributed computation. We consider the well-studied point-to-point communication model, where we are given a complete network of n nodes (processors) with distinct identifiers (we assume $O(\log n)$-size *id*s) and each node knows the (nonnegative) weights associated with its incident edges (bidirectional communication links) but not the identifiers of its neighbors (see e.g., [31, 21]). The communication between any two nodes happens by sending/receiving messages along the edge between them and all nodes perform the same algorithm. We assume that $O(\log n)$ bits can be transferred in one step per edge and a node can send messages through all its incident links at the same time (see e.g., [31]).

The following distributed algorithm, in Figure 2, is a realization of the Random NN-scheme in a distributed complete network. Here, each node chooses its rank by choosing a number uniformly and independently at random from $[0, 1]$.[1] Then each node, in rounds, keeps sending FIND messages to its neighbors beginning with the nearest neighbor, in non-decreasing order of the edge weights, until it receives k ACCEPT messages. The FIND messages contain the sender's random number (chosen from $[0, 1]$) and *id*. The receiver of a FIND message compares its rank with the rank of the sender. If the receiver's rank is higher than the sender's rank, the receiver sends an ACCEPT message back to the sender of the FIND message. Note that we do not make any assumption about the weights of the edges in designing the distributed algorithm and in analyzing its time and message complexity. However, as we have seen in the previous section, the quality (the weight) of the k-connected subgraph constructed by this algorithm, with respect to the quality of the optimal k-connected subgraph, depends on the properties satisfied by the weights of the edges.

Message and Time Complexity. It is interesting to analyze the message complexity and the time complexity, and their tradeoffs in the distributed model we consider (i.e., point to point communications with all processors forming a clique). A naive method for finding the k nearest higher ranked nodes is: each node probes one neighbor at a time, to find the ranks of its neighbors, in

[1] The ranks can be also chosen uniformly from, say, $[1, n^4]$ and the ranks will be unique with high probability. Or, as is done in the algorithm, we assume that each node has a unique label which is used to break the ties. This does not alter any of our proofs or the results.

Distributed k-connected graph algorithm

Input: A complete graph $\mathbb{K}_n = G(V, E)$. We assume each node has a unique id from a totally ordered set.
Output: A k-connected subgraph G_k. On termination, each node knows which of its adjacent edges are in G_k.

Each node $u \in V$ executes the following protocol independently and simultaneously:

1. Choose the rank $r(u)$ as follows: generate a random number $p(u) \in [0, 1]$. We say $r(v) > r(u)$ if and only if [$p(v) > p(u)$] or [$p(v) = p(u)$ and $id(v) > id(u)$].
2. Find $|\eta(u)|$ nearest nodes w with $r(w) > r(u)$, and add the edges (u, w) to G_k. Find the w's as follows:
   $t \leftarrow 1$ ▶ t `is the round number`
   REPEAT ▶ `A FIND message includes` $p(u)$ `and` $id(u)$
   If $t = 1$, u sends FIND messages to all $v \in \Gamma_u(k)$ simultaneously;
   If $t \geq 2$, u sends FIND messages to all $v \in [\Gamma_u(2^{t-1}k) - \Gamma_u(2^{t-2}k)]$ simultaneously;
   $t \leftarrow t + 1$
   UNTIL u received k ACCEPT messages or probed all of its neighbors.
3. Upon receipt of a FIND message from any v, send back an ACCEPT message to v iff $r(u) > r(v)$.

Figure 2: Distributed implementation of the Random-NN scheme.

nondecreasing order of edge weights. By Lemma 3.2, the expected number of the messages each node needs to exchange is $O(k \log \frac{n}{k})$ to find the k higher ranked nodes (Note that a node made its k^{th} connection means that it already made all the required connections). This gives an expected total of $O(kn \log \frac{n}{k})$ messages. However, the time complexity of this implementation is $\Theta(n)$ since there will be a node (the highest ranked node) which has to probe all its $(n-1)$ neighbors. On the other hand, if we want to get a better time complexity at the expense of more messages, consider a different protocol: each node sends its rank (the random number and the *id*) to all its neighbors in one step (one round); this finishes in $O(1)$ time, but consumes $\Theta(n^2)$ messages.

To reduce both the time complexity and the message complexity, we consider the *hybrid* protocol given in Figure 2, where in the first round, a node probes the first k nearest neighbors and in the subsequent rounds $t \geq 2$, it probes the next $2^{t-2}k$ nearest neighbors until it succeeds in finding the k nearest higher ranked neighbors. Below we present the analysis of the time and message complexity of this protocol.

Theorem 5.1 *The protocol of Figure 2 takes $O(\lg \frac{n}{k})$ time and uses expected $O(kn \lg \frac{n}{k})$ messages.*

Proof: A node u needs $1 + \lceil \lg \frac{n-1}{k} \rceil$ rounds to probe all of its $n-1$ neighbors. Therefore, the protocol takes at most $1 + \lceil \lg \frac{n-1}{k} \rceil \leq 2 + \lg \frac{n}{k}$ time. To bound the message complexity, we calculate the expected number of the messages a node sends before it finds the k neighbors of higher ranks.

In the t^{th} round for $t \geq 2$, a node u sends FIND messages to all nodes in $\Gamma_u(2^{t-1}k) - \Gamma_u(2^{t-2}k)$. Using Lemma 3.1, the probability that u makes the k^{th} connection in the round t is

$$\sum_{i=2^{t-2}k+1}^{2^{t-1}k} \frac{k}{i(i+1)} = k \left\{ \frac{1}{2^{t-2}k+1} - \frac{1}{2^{t-1}k+1} \right\}$$

$$= k \left\{ \frac{2}{2^{t-1}k+2} - \frac{1}{2^{t-1}k+1} \right\} \leq \frac{k}{2^{t-1}k+1} \leq \frac{1}{2^{t-1}}.$$

Notice that the above upper bound for the probability can also be used for $t = 1$ as $1/2^{t-1}$ evaluates to 1 when $t = 1$. The number of SEND messages u sends in the first t rounds is $2^{t-1}k$. Thus, the expected number of SEND messages by u is at most

$$\sum_{t=1}^{1+\lceil \lg \frac{n-1}{k} \rceil} (2^{t-1}k)\frac{1}{2^{t-1}} \leq 2k + k \lg \frac{n}{k}.$$

Moreover, u receives at most k ACCEPT messages. Thus, using linearity of expectation for n nodes, the expected total number of the messages is $3kn + kn \lg \frac{n}{k}$. □

Remarks. **1.** In the distributed model we consider (i.e., point to point communication with all processors forming a clique), a modification of the proof given by Korach, Moran, and Zaks in [21] (which was given for MST) shows a lower bound of $\Omega(n^2)$ on the number of the messages needed to construct an optimal k-connected spanning subgraph (for any $1 \leq k \leq \lfloor n/2 \rfloor - 1$) in a complete weighted metric graph; this lower bound is independent of the length of the messages. Thus, in general, the expected message complexity of our randomized algorithm is significantly better than the deterministic lower bound. Also, the message complexity of our algorithm is optimal in the sense that $\Omega(n \log n)$ is a lower bound on the number of the messages needed to construct any spanning tree [22]. A lot of work had been devoted to finding spanning tree (equivalent to leader election) algorithms having $O(n \log n)$ message complexity in this model (see e.g.,[22, 1, 17]) and our protocol also gives a very simple spanning tree and leader-election protocol that has $O(n \log n)$ (expected) message complexity.
2. It is also quite easy to adapt the above algorithm for a "broadcast" setting which is a typical model for wireless networks (see e.g., [27]). In such a setting, nodes are assumed to be in a geometric space (e.g., a plane) and a node communicates with its neighbors by broadcasting a message. All nodes within the broadcast range can receive the message (ignoring collisions). To implement our algorithm, a node has to progressively increase its broadcast range (in a similar doubling fashion) till it finds the nearest nodes of higher ranks. We analyze such a strategy in detail in a separate paper which also contains experimental results in the context of the wireless sensor networks. [20].

6 Conclusion and Further Work

We showed and analyzed a simple randomized approximation scheme for constructing a low-weight k-connected spanning subgraph. We also presented its efficient implementation in a complete network of processors. The proposed algorithm has low time and message complexity while giving a relatively good approximation ratio for the metric graphs, random geometric graphs, and random edge-weight graphs. It is interesting to see whether the ideas in this paper can be used to design an efficient distributed algorithm for the more challenging problem of finding a k-connected subgraph in an arbitrary general graph (need not be complete). The local nature of the NN-scheme seems suitable for designing a simple and efficient *dynamic* algorithm (especially in a distributed setting), where the goal is to maintain a k-connected graph of good quality, as nodes are added or deleted. This looks promising for future work.

References

[1] Y. Afek and E. Gafni. Simple and efficient distributed algorithms for election in complete networks. In *Proc. 22nd Ann. Allerton Conference on Communication, Control, and Computing*, pages 689–698, 1984.

[2] J. Cheriyan, M. Kao, and R. Thurimella. Scan-first search and sparse certificates: an improved parallel algorithm for k-connectivity. *SIAM Journal of Computing*, 22(1):157–174, 1993.

[3] J. Cheriyan, S. Vempala, and A. Vetta. Approximation algorithms for minimum-cost k-vertex connected subgraph. In *34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 206–312, 2002.

[4] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[5] A. Czumaj and A. Lingas. On approximability of the minimum-cost k-connected spanning subgraph problem. In *Proceedings of 10th ACM-SIAM Symposium on Discrete Algorithms*, pages 74–83, 2002.

[6] J. Edmonds. Edge-disjoint branchings. *Combinatorial Algorithms, R. Rustin Ed., Academic Press, New York*, pages 91–96, 1973.

[7] M. Elkin. Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problem. In *Proceedings of Symposium on Theory of Computing (STOC)*, June 2004.

[8] J. Fill and M. Steele. Exact expectations of minimal spanning trees for graphs with random edge weights. In *Proceedings of the Symposium "Stein's Method and Applications: A Program in Honor of Charles Stein"*, 2004.

[9] A. Frank. Connectivity and network flows. *Survery Chapter in Handbook of Combinatorics, Eds. R. Graham, M. Grotschel and L. Lovasz, Elsevier Science B.V.*, pages 111–177, 1995.

[10] A. Frieze. On the value of a random minimum spanning tree problem. *Discrete Applied Mathematics*, 10(1):47–56, 1985.

[11] R. Gallager, P. Humblet, and P. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems*, 5(1):66–77, January 1983.

[12] D. Gamarnik. The expected value of random minimal spanning tree of a complete graph. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, January 2005.

[13] R. Graham, D. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science, Second Edition*. Addison-Wesley Publishing Company, Inc., 1989.

[14] D. Hochbaum. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, Boston, MA, 1996.

[15] W. Hohberg. How to find biconnected components in distributed networks. *Journal of Parallel and Distributed Computing*, 9(4):374–386, 1990.

[16] S. Huang. A new distributed algorithm for the biconnectivity problem. In *Proceedings of International Conference on Parallel Processing*, volume III, pages 106–103, 1989.

[17] P. Humblet. Selecting a leader in a clique in $o(n \log n)$ messages. In *Proc. 23rd conf. on decision and control*, pages 1139–1140, 1984.

[18] M. Imase and B.M. Waxman. Dynamic steiner tree problem. *Siam J. Discrete Math*, 4(3):369–384, 1991.

[19] E. Jennings and L. Motyckova. Distributed algorithms for sparse k-connectivity certificates. In *Proceedings of the Symposium on Principles of Distributed Computing (PODC)*, page 180, 1996.

[20] M. Khan, V.S.A. Kumar, and G. Pandurangan. Local algorithms for constructing approximate minimum spanning trees with applications to wireless sensor networks. Technical report, Department of Computer Science, Purdue University, 2005. http://www.cs.purdue.edu/homes/gopal/localapproxmst.pdf.

[21] E. Korach, S. Moran, and S. Zaks. The optimality of distributive constructions of minimum weight

and degree restricted spanning trees in a complete network of processors. *SIAM Journal of Computing*, 16(2):231–236, 1987.

[22] E. Korach, S. Moran, and S. Zaks. Optimal lower bounds for some distributed algorithms for a complete network of processors. *Theoretical Computer Science*, 64:125–132, 1989. Conference version: Proceedings of the ACM Symposium on the Principles of Distributed Computing (PODC) 1984.

[23] G. Kortsarz and Z. Nutov. Approximating node connectivity problem via set covers. In *3rd International workshop on approximation algorithms for combinatorial optimization (APPROX)*, pages 194–205, 2000.

[24] G. Kortsarz and Z. Nutov. Approximation algorithm for k-node connected subgraphs via critical graphs. In *36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 206–312, 2004.

[25] G. Kortsarz and Z. Nutov. Approximating min-cost connectivity problems. *Survey Chapter in Handbook on Approximation Algorithms an Metaheuristics, Ed. T. F. Gonzalez, Chapman & Hall*, 2006.

[26] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally. In *Proceedings of the Symposium on Principles of Distributed Computing (PODC)*, 2004.

[27] X. Li. Algorithmic, geometric and graphs issues in wireless networks. *Journal of Wireless Communications and Mobile Computing (WCMC)*, 3(2):119–140, 2003.

[28] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithm and Probabilistic Analysis*. Cambridge University Press, first edition, 2005.

[29] A. Panconesi and R. Rizzi. Some simple distributed algorithms for sparse networks. *Distributed Computing*, 14(2):97–100, 2001.

[30] A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. *SIAM Journal on Computing*, 26:350–368, 1997.

[31] D. Peleg. *Distributed Computing: A Locality Sensitive Approach*. SIAM, 2000.

[32] R. Rajaraman. Topology control and routing in ad hoc networks: A survey. *SIGACT News*, 33:60–73, 2002.

[33] D. Rosenkrantz, R. Stearns, and P. Lewis. An analysis of several heuristics for the traveling salesman problem. *SIAM J. Comput.*, 6(3):563–581, 1977.

[34] M. Steele. Asymptotics for euclidian minimal spanning trees on random points. *Probability Theory and Related Fields*, 92:247–258, 1992.

[35] R. Thurimella. Sub-linear distributed algorithms for sparse certificates and biconnected components (extended abstract). In *Symposium on Principles of Distributed Computing (PODC)*, pages 28–37, 1995.

[36] Z. Toroczkai and K. Bassler. Jamming is limited in scale-free systems. *Nature*, 428:716, 2004.