A Faster Distributed Protocol for Constructing a Minimum Spanning Tree
(Extended abstract)

Michael Elkin [*][†]

Abstract

This paper studies the problem of constructing a *minimum-weight spanning tree (MST)* in a distributed network. This is one of the most important problems in the area of distributed computing. There is a long line of gradually improving protocols for this problem, and the state of the art today is a protocol with running time $O(\Lambda(G) + \sqrt{n} \cdot \log^* n)$ due to Kutten and Peleg [KP95], where $\Lambda(G)$ denotes the diameter of the graph G. Peleg and Rubinovich [PR99] have shown that $\tilde{\Omega}(\sqrt{n})$ time is required for constructing MST even on graphs of small diameter, and claimed that their result "establishes the asymptotic near-optimality" of the protocol of [KP95].

In this paper we refine this claim, and devise a protocol that constructs the MST in $\tilde{O}(\mu(G,\omega) + \sqrt{n})$ rounds, where $\mu(G,\omega)$ is the MST-radius of the graph. The ratio between the diameter and the MST-radius may be as large as $\Theta(n)$, and, consequently, on some inputs our protocol is faster than the protocol of [KP95] by a factor of $\tilde{\Omega}(\sqrt{n})$. Also, on every input, the running time of our protocol is never greater than twice the running time of the protocol of [KP95].

As part of our protocol for constructing an MST, we develop a protocol for constructing neighborhood covers with a drastically improved running time. The latter result may be of independent interest.

1 Introduction

1.1 Distributed Computing Consider a weighted undirected n-vertex graph $(G = (V, E), \omega)$, with a non-negative weight function ω. Suppose that every vertex hosts a processor with *unbounded computational power*, but with *limited initial knowledge*. Specifically, assume that each vertex (the terms "vertex" and "processor" are synonyms in this context) is attached a distinct identity number from the set $\{1, 2, \ldots, n\}$, and at the beginning of the computation each vertex v accepts as input its own identity number, the identity numbers of its neighbors in G, and the weights of the edges that are adjacent to v. The vertex may also accept some additional inputs as specified by the problem at hand.

The vertices are allowed to communicate through the edges of the graph G. The communication is synchronous, and occurs in discrete pulses, called *rounds.* In particular, all the vertices wake up simultaneously at the beginning of round 1, and from this point on the vertices always know the number of the current round. On each round each vertex v is allowed to send an arbitrary message of size $O(\log n)$ through each edge $e = (v, u)$ that is adjacent to v, and the message will arrive to u at the end of the current round. The weights of the edges are at most polynomial in the number of vertices n, and, therefore, a weight of a single edge can be communicated in one round. We will refer to this model of distributed computation as the $\mathcal{CONGEST}$ model. We will also consider the so-called $\mathcal{LOCAL}$ model, in which messages of *unbounded size* can be delivered through an edge in one round. Finally, in-between there is the $\mathcal{CONGEST}(B)$ model in which the upper bound on the size of the messages is B, and B is a parameter.

There are several measures of efficiency of distributed algorithms (also called "protocols"), but we will concentrate on one of them, specifically, the *running time*, that is, the number of rounds of distributed communication. Note that the computation that is performed by the vertices *locally* is "free", i.e., it does not affect the time efficiency measure.

This model of computation is often termed as *distributed computing with no shared memory*, or, shortly, *distributed computing.* The distributed computing model has been attracting a lot of research attention during last two decades (e.g., [Awe85, Awe87, CT85, Gaf85, AP90a, Lin92, LS91, ABCP93, ABCP96, PR99, Elk01]; see also [Pel00] and the references therein). The importance of the model stems from both practical and theoretical considerations. From a practical perspective the model serves as a fairly good abstraction of today's most prominent computer networks, particularly the Internet. From a theoretical perspective the model can

[*]Department of Computer Science, Yale University, New Haven, CT, USA, 06520-8285. `elkin@cs.yale.edu`

[†]Part of this work was done in School of Mathematics, Institute for Advanced Study, Princeton, NJ, USA, 08540.

be seen as a gross generalization of the communication complexity model, in which the graph G consists of a single edge.

1.2 Distributed *MST* Problem: Previous Research The *distributed minimum-weight spanning tree* (henceforth, *MST*) problem is one of the most important problems in the area of distributed computing. In this problem each vertex v should return as output the set of edges that are adjacent to v and belong to the *MST* of the graph (assuming that the latter is unique; otherwise, the outputs of all the vertices should be consistent, and form an *MST*). The study of the *MST* problem was initiated twenty years ago in a seminal paper by Gallager et al. [GHS83], that devised a protocol that constructs the *MST* in $O(n \cdot \log n)$ rounds of distributed computation in the $\mathcal{CONGEST}$ model. This result was improved soon afterwards by Chin and Ting [CT85] to $O(n \cdot \log\log n)$, further improved to $O(n \cdot \log^* n)$ by Gafni [Gaf85], and consequently improved by Awerbuch [Awe87] to an *existentially optimal* running time of $O(n)$.

In the late eighties it was observed that for most existing networks G, their diameter $\Lambda(G)$ is significantly smaller than the number of vertices n, and that therefore it is desirable to design protocols whose running time is bounded in terms of $\Lambda(G)$ rather than in terms of n [Awe89]. The first such protocol for the *MST* problem was designed by Garay et al. [GKP93], and its running time is $O(\Lambda(G) + n^{0.61})$. The result was later improved by Kutten and Peleg [KP95] to $O(\Lambda(G) + \sqrt{n} \cdot \log^* n)$, and this is the state of the art today. Table 1 summarizes the upper bounds on the time complexity of the *MST* problem in the $\mathcal{CONGEST}$ model. The lack of progress in improving the result of [KP95] led researchers to work on lower bounds. Peleg and Rubinovich [PR99] have shown that $\Omega(\frac{\sqrt{n}}{\log n})$ rounds of distributed computation are required for constructing the *MST* even when the input graphs have low diameter. Elkin [Elk03] has recently improved this lower bound to $\Omega(\sqrt{\frac{n}{\log n}})$.

The upper bound of Kutten and Peleg [KP95] in conjunction with the lower bound of Peleg and Rubinovich [PR99] may leave the impression that there is no more room for a significant improvement in our understanding of the complexity of the *MST* problem. Indeed, Peleg and Rubinovich explicitly stated in [PR99] that in view of their lower bound, the protocol of [KP95] is asymptotically near-optimal. However, an indication that the protocol of [KP95] can still be improved was provided by Peleg in [Pel00a], that devised a protocol that constructs the *MST* in $O(\gamma(G))$ rounds, where

Running time	Reference
$O(n \cdot \log n)$	Gallager et al. [GHS83]
$O(n \cdot \log\log n)$	Chin, Ting FOCS'85 [CT85]
$O(n \cdot \log^* n)$	Gafni PODC'85 [Gaf85]
$O(n)$	Awerbuch STOC'87 [Awe87]
$O(\Lambda(G) + n^{0.61})$	Garay et al. FOCS'93 [GKP93]
$O(\Lambda(G)+$ $\sqrt{n} \cdot \log^* n)$	Kutten, Peleg PODC'95 [KP95]
$\tilde{O}(\mu(G,\omega) + \sqrt{n})$	This paper.[1]

Table 1: The summary of upper bounds on the complexity of the *MST* problem.

$\gamma(G)$ denotes the *cyclic radius* of the graph G, in the $\mathcal{LOCAL}$ model. However, this result does not apply to the more standard $\mathcal{CONGEST}$ model, while all the aforementioned results do. The definition of the cyclic radius $\gamma(G)$ is deferred to Section 2, and meanwhile we remark that for every graph G, $\gamma(G) \le \Lambda(G)$, and it is often the case that $\gamma(G) \ll \Lambda(G)$.

We also remark that the protocol of [Pel00a] does not detect termination. Intuitively, it means that if an adversary is allowed to stop the execution of the protocol and to force the vertices to return an output in an arbitrary moment, they would be able to return the *MST* of the input graph if this termination happens after $\Omega(\gamma(G))$ rounds.

1.3 Our Results Upper Bounds: The main motivation for the result of [Pel00a] was the quest for the "correct" graph parameter that reflects the complexity of the *MST* problem to as large extent as possible. It was suggested in [Pel00a] that the cyclic radius could be such a parameter. In this paper we prove that it is not true, and single out another parameter, specifically, the *MST-radius* $\mu(G,\omega)$, that also serves as an upper bound on the complexity of the *MST* problem in the $\mathcal{LOCAL}$ model. The *MST*-radius is never greater than the cyclic radius, and the gap between the two parameters may be arbitrarily large; in fact, we show an example of an infinite family of n-vertex graphs with cyclic radius equal to $n/2$ and *MST*-radius equal to 1! (The definition of *MST*-radius is deferred to Section 2.)

[1]We use the notations $\tilde{O}(f(n))$ and $\tilde{\Omega}(f(n))$ to denote $O(f(n) \cdot \text{polylog}(f(n)))$ and $\Omega(f(n)/\text{polylog}(f(n)))$, respectively.

Moreover, after eight years that witnessed no progress on upper bounds for the *MST* problem in the $\mathcal{CONGEST}$ model, we devise a randomized protocol for the problem with worst-case running time $\tilde{O}(\mu(G,\omega)+\sqrt{n})$. Since for many graphs (G,ω) there is a large gap between the diameter $\Lambda(G)$ and the *MST*-radius $\mu(G,\omega)$, it is often the case that our protocol is far more efficient than the previously best-known protocol [KP95]. Specifically, the ratio between the running times of these protocols may be as large[2] as $\tilde{\Omega}(\sqrt{n})$. Also, on any input, our protocol is as efficient as the protocol of [KP95], up to a polylogarithmic factor in n (and, actually, the two protocols can be combined in the obvious way so that the running time of the combined protocol is always at most twice the minimum of the two running times).

We remark that it is natural that there exist graphs on which the upper bound on the running time of our protocol is no better than the running time of the protocol of [KP95]. Indeed, this is the case also when comparing the running time of the protocol of [KP95] with that of [GKP93], or with that of [Awe87]. In other words, the *existential optimality* of the protocol of [Awe87] rules out the possibility of devising a protocol that has strictly smaller running time than the protocol of [Awe87] on *every* graph.

However, unlike the previous protocols [KP95, GKP93, Awe87], in our protocol the vertices should accept the *MST*-radius $\mu(G,\omega)$ of the input graph as an additional input (or, else, the protocol does not detect termination). To weaken this assumption we generalize the protocol to work in the scenario when each vertex v in the graph accepts as input its own upper bound $\hat{\mu}_v$ of the *MST*-radius of the graph, and for different vertices these upper bounds are allowed to disagree. Denoting the maximum of these upper bounds by $\hat{\mu}$, the running time of our protocol in this scenario is $\tilde{O}(\hat{\mu}+\sqrt{n})$.

We remark that *some* assumption of this sort is necessary, as protocol with our running time in which the vertices accept no additional input provably provides wrong outputs on some inputs. Also, the assumption that all the vertices in the graph know an estimate of some parameter of the input graph, such as the number of vertices or the diameter of the graph, is quite common in distributed computing [LS91, AGLP89], especially for protocols whose running time is $o(\Lambda(G))$ (protocols that run $\Omega(\Lambda(G))$ rounds can compute these parameters within their complexity bounds).

[2]Moreover, both protocols generalize to the $\mathcal{CONGEST}(B)$ model. The running time of the generalization of the protocol of [KP95] is $O(\Lambda(G)+\sqrt{(n\log n)/B}\log^* n)$, and the running time of our protocol is $\tilde{O}(\mu(G,\omega)+\sqrt{n/B})$. Hence, the maximum ratio between the running times of the two protocols grows with B. Specifically, it is $\min\{n, \sqrt{n\cdot B}/\log^3 n\}$.

Lower Bounds: We also show that the *MST*-radius "deserves its name", that is, reflects the complexity of the *MST* problem in several senses (and, in particular, it is rather unlikely that a protocol for constructing an *MST* with running time smaller than the *MST*-radius exists even in the $\mathcal{LOCAL}$ model). All our lower bounds are proved in the $\mathcal{LOCAL}$ model, and therefore they obviously apply to the $\mathcal{CONGEST}$ model as well.

First, we consider the class of *coarsening* protocols that do not detect termination. Intuitively, a protocol is coarsening if for every input its output is always a superset of the correct answer of the problem on this input, even if the protocol is stopped prematurely. We show that for *every* coarsening protocol and *every* input (G,ω) of the *MST* problem, the running time of the protocol is at least $\mu(G,\omega)$, or else the protocol returns a wrong output on some other instance. Note that this lower bound is matched *exactly* by our protocol for constructing the *MST* in the $\mathcal{LOCAL}$ model.

However, note that both these upper and lower bounds refer to the distributed network as a blackbox, considering the running time of a protocol as a whole, and making no distinction between the individual vertices. While this is the standard way to think about the running time of a protocol (as maximum of the running times of individual vertices that run the protocol), in this paper we suggest a refined way of defining the running time of distributed protocols. Specifically, we introduce the notion of *vector running time* of a distributed protocol, which is the n-tuple of the running times of different vertices. We also refine the notion of the *MST*-radius, and define the *vector MST-radius*, $\vec{\mu}(G,\omega)$, of a graph. The vector *MST*-radius of a graph (G,ω) is an n-dimensional vector indexed by the vertex set V of G, and for every vertex $v\in V$ the vth coordinate of $\vec{\mu}(G,\omega)$, denoted $\mu_v(G,\omega)$, is a parameter that depends both on the graph (G,ω) and on the vertex v (see Section 2).

We show that for *every* coarsening protocol and *every* input (G,ω) of the *MST* problem, and *every* vertex v of G, the running time of the vertex v in the protocol is at least the vth coordinate of the *MST*-radius $\vec{\mu}(G,\omega)$, that is, $\mu_v(G,\omega)$. This lower bound is also exactly matched by protocol for constructing the *MST* in the $\mathcal{LOCAL}$ model, i.e., when the protocol is run on the graph (G,ω), for every vertex v of the graph, the running time of the restriction of this protocol to v is precisely $\mu_v(G,\omega)$. In terms of our refined notion of running time, the *vector* running time of our protocol is precisely equal to the vector *MST*-radius of the graph.

Second, we consider the MST^* problem, a variant of the MST problem in which every vertex accepts the MST-radius of the input graph (G, ω) as part of its input. We show that for *every* graph (G, ω), every protocol that solves the MST^* problem on (G, ω) requires $\Omega(\mu(G, \omega))$ rounds even in the $\mathcal{LOCAL}$ model, and this lower bound is matched up to a constant by our upper bound for the $\mathcal{LOCAL}$ model.

1.4 Every-Case Complexity While these lower bounds do not rule out completely the possibility that our protocol would ever be improved, they can still serve as a strong evidence that the MST-radius is not an arbitrary parameter, and that it reflects in a rather strong sense the complexity of the MST problem.

It should be noticed that in the $\mathcal{LOCAL}$ model our upper and lower bounds agree on *every instance.* This *every-case* or *instance* optimality is a rather rare phenomenon, that, however, was already observed more or less explicitly in several different contexts.

First, more than thirty years ago, Levin [Lev72] proved that for every NP-relation there exists an optimal search algorithm. Without going into details of the precise sense in which this algorithm is optimal, we note that this "optimality" is achieved on *every* instance of the appropriate search problem. More recently, Dwork and Moses [DM90] have devised an algorithm for Byzantine agreement, and their algorithm is optimal for *every* sequence of crash failures of the participating processors. Very recently, Fagin et al. [FLN01] have devised an aggregation algorithm for database systems, and have shown that their algorithm is optimal up to a factor of 2 on *every* instance of the problem, as far as the attention is restricted to a certain rather wide class of algorithms. Garay et al. [GKP93] were probably the first to discuss the general phenomenon of algorithmic optimality on *every* instance explicitly, and they called it *universal optimality.* Fagin et al. [FLN01] further formalized this notion of optimality, and termed it *instance-optimality.*

We take the discussion on instance-optimality one step forward, and define a *new complexity measure of problems* that has the property that instance-optimal algorithms are optimal with respect to this measure. We call this measure *every-case complexity*, and interpret our results concerning the MST problem in the $\mathcal{LOCAL}$ model in terms of this new complexity measure. Specifically, the aforementioned result concerning the MST^* problem means that the every-case complexity of this problem is (up to a constant factor) equal to the MST-radius. We also generalize the notion of every-case complexity measure to adapt it to the distributed model with the *vector* notion of running time. For this scenario we introduce the *vector every-case complexity measure* of problems, and in terms of this measure our result concerning the coarsening will-maintaining protocols for the MST problem in the $\mathcal{LOCAL}$ model means that the vector every-case complexity of the MST problem with respect to an appropriate class of protocols is precisely equal to the vector MST-radius. (Note that both the complexity measure and the vector MST-radius are vector functions with common domain; the latter is the class of all weighted graphs.)

We remark that these semantic interpretations do not aim to shed a new light on the complexity of the MST problem, but rather to illustrate that the scalar and vector every-case complexity measures, introduced in this paper are meaningful, and convey more information about the complexity of certain problems than the classic worst-case and average-case complexity measures.

Our Techniques: We believe that our main contribution is in realizing that the result of Kutten and Peleg [KP95] still leaves a large room for improvement. From a technical perspective, the idea that makes possible such an improvement is to use *neighborhood covers* for constructing the MST. Neighborhood covers were introduced in [ABCP93], and were found very useful for various applications. In particular, they were used in [AP90a] for network synchronization, in [AP92] for routing, and in [ABCP93, Elk01] for computing almost shortest paths and constructing spanners. However, so far neighborhood covers were not used for constructing the MST, and moreover, all the existing distributed protocols for constructing neighborhood covers that apply for the entire range of the parameters either require super-linear time [ABCP93, AP90b], or apply only to the $\mathcal{LOCAL}$ model [ABCP96]. This running time exceeds by far our desired resource limits, that is, a running time of $\tilde{O}(\mu + \sqrt{n})$. To overcome this difficulty we devise a new efficient randomized protocol for constructing neighborhood covers.

Our protocol for constructing neighborhood covers is based on Cohen's parallel algorithm for constructing *pairwise covers* [Coh93], and adapts some techniques from [ABCP96]. There are two major differences between our protocol and Cohen's algorithm. First, pairwise covers are very related but yet different from neighborhood covers. The more striking is the difference between the parallel and distributed models of computation. In the parallel model the objective is to minimize the maximum number of steps of *computation* that are performed by each processor, and the number of processors involved, while in the distributed model the objective is to minimize the number of rounds of *communica-*

tion between the processors that have *unbounded computational power*. Also, the algorithm of [Coh93] uses the assumption that the processors have shared memory, and this is not the case in the distributed model.

We believe that our distributed protocol for constructing neighborhood covers is of independent interest. To illustrate its applicability, we show that it immediately gives rise to a drastically improved protocol for constructing sparse spanners (the running time of our protocol is by a factor of $n/(\text{polylog}(n))$ smaller than the running time of the previously best-known protocol [ABCP93] for this task). We believe that other applications of our efficient protocol for constructing neighborhood covers will be found in future.

Remark: Although our protocol for constructing the covers, and, consequently, the *MST*, is randomized[3], there is a hope to derandomize it while maintaining similar running time. Specifically, Awerbuch et al. [ABCP96] devised a *deterministic* protocol for constructing neighborhood covers with running time greater by a factor of $2^{O(\sqrt{\log n})}$ than the running time of our randomized protocol. Unfortunately, their result applies only to the $\mathcal{LOCAL}$ model of distributed computation. So far we were not able to adapt their protocol to the $\mathcal{CONGEST}$ model, but we believe that such adaptation should be possible. If the latter turns out to be true, it will give rise to a *deterministic* protocol for constructing the *MST* in $\mu(G,\omega) \cdot 2^{O(\sqrt{\log n})} + \tilde{O}(\sqrt{n})$ rounds.

Structure of the paper: The simplest form of our protocol for constructing the *MST* and its analysis appear in Section 3. Its generalization to the scenario where the upper bounds on the *MST*-radius that the vertices accept as input are allowed to disagree appears in Section 4. Due to space limitations, several sections of this paper are omitted from this extended abstract. One of the omitted sections describes our protocol for constructing neighborhood covers, another section is devoted to the notions of scalar and vector every-case complexity, and one more section presents our results regarding the *MST* problem in the $\mathcal{LOCAL}$ model of distributed computation. Also, most proofs are omitted. The full version of this paper can be found at http://www.math.ias.edu/˜ elkin.

2 Preliminaries

Without loss of generality, we assume that the graph is connected. We also assume that all the weights of edges in the graphs are distinct. It implies, in particular, that the *MST* of the graph is unique. The definitions and the results generalize readily to the case when the weights are not necessarily distinct.

For a vertex $u \in V$ and an edge $e \in E$, let $dist_G(u,e) = \min\{dist_G(u,v) \mid v \in e\}$. For a vertex $u \in V$ and a cycle C, let the *vertex-edge radius* of the cycle C with respect to vertex u, denoted $VxEdgRad_u(C)$, be $\max\{dist_G(u,e') \mid e' \in E(C)\}$. For an edge e, let $\mathcal{Z}(e)$ be the set of cycles C in which the edge e is the heaviest edge. For a vertex $u \in V$ and an edge $e \in E$ that is adjacent to u, let $VxEdgElimRad(u,e) = \min\{VxEdgRad_u(C) \mid C \in \mathcal{Z}(e)\}$. (The minimum of an empty set is defined as zero.) Intuitively, $VxEdgElimRad(u,e)$ is the number of rounds required to the vertex u to discover that the edge e that is adjacent to u does not belong to the *MST*. The *MST-radius* of a weighted graph $(G=(V,E),\omega)$ with respect to a vertex $u \in V$, denoted $\mu_u(G,\omega)$, is defined by $\mu_u(G,\omega) = \max\{VxEdgElimRad(u,e) \mid e \in E, u \in e\}$. The *MST-radius* of the graph $(G=(V,E),\omega)$ with vertex set $V=\{1,2,\ldots,n\}$ is the vector $\vec{\mu}(G,\omega) = (\mu_1(G,\omega),\mu_2(G,\omega),\ldots,\mu_n(G,\omega))$, and the *maximum MST-radius* of the graph (G,ω), denoted $\mu(G,\omega)$, is the ℓ_∞-norm of the vector $\vec{\mu}(G,\omega)$, i.e., $\mu(G,\omega) = \max\{\mu_u(G,\omega) \mid u \in V\}$.

For a vertex $u \in V$ and a cycle C, the *vertex radius* of C with respect to u is $VxRad_u(C) = \max\{dist_G(u,w) \mid w \in V(C)\}$. The *vertex radius* of the cycle C is $\min\{VxRad_u(C) \mid u \in V\}$. The *cyclic radius* of the graph G, denoted $\gamma(G)$, is the maximum of the vertex radii of the cycles of G.

To conclude the discussion about the parameters that affect the every-case complexity of the *MST* problem, we show an example of a graph with an *MST*-radius of 1, and a cyclic radius of $n/2$. This example shows that the gap between the cyclic radius and the *MST*-radius can be arbitrarily large.

Consider a $(2k)$-vertex ladder $G=(V,E)$ with vertex set $V = U \cup W$, $U=\{u_1,u_2,\ldots,u_k\}$, $W=\{w_1,w_2,\ldots,w_k\}$, and edge set $E = E_1 \cup E_2$, $E_1 = \{(u_i,u_{i+1}),(w_i,w_{i+1}) \mid i=1,2,\ldots,k-1\}$, $E_2 = \{e_i = (u_i,w_i) \mid i=1,2,\ldots,k\}$, with weight function ω defined by $\omega(e)=0$ for each $e \in E_1$, and $\omega(e_i)=i$ for $i=1,2,\ldots,k$. Observe that any cycle C contains at least one edge from the set $\{e_i \mid i=2,3,\ldots,k\}$. Observe that for any edge $e \in E_1 \cup \{e_1\}$, there is no cycle C in the graph such that e is the heaviest edge in C. Hence, for each edge $e \in E_1 \cup \{e_1\}$ and any endpoint $x \in e$, $VxEdgElimRad(x,e)=0$. For each $i=2,3,\ldots,k$, $VxEdgElimRad(u_i,e_i) = VxEdgElimRad(w_i,e_i) = 1$. Hence, $\mu_{u_1}(G,\omega) = \mu_{w_1}(G,\omega) = 0$, and $\mu_{u_i}(G,\omega) = \mu_{w_i}(G,\omega) = 1$, for $i=2,3,\ldots,k$. Hence the maximum *MST*-radius,

[3]With negligible probability over the coin tosses the protocol may output a cyclic subgraph of the input graph.

$\mu(G,\omega)$, is equal to 1. On the other hand, all the vertex radii of the cycle $(u_1, u_2, \ldots, u_k, w_k, w_{k-1}, \ldots, w_1, u_1)$ are equal to k. Hence, the cyclic radius of this graph is equal to $\frac{n}{2}$.

3 The $\mathcal{CONGEST}$ Model

We next describe our protocols for constructing the *MST* in the $\mathcal{CONGEST}(B)$ model.

3.1 Overview The protocol of [KP95] consists of two parts. The first part constructs an *MST*-forest, that is, a forest of subtrees (also called *fragments*) of the *MST*. The second part "filters" the inter-fragment edges, and leaves only those that belong to the *MST*. This filtering is performed by a pipelined convergecast of these edges through a BFS tree of the graph, and, thus, requires at least $\Lambda(G)$ rounds (as the depth of the BFS tree is $\Lambda(G)$). Our protocol has a similar structure to that of [KP95], and its first part also involves constructing an *MST*-forest, though with slightly different parameters. However, in its second part, instead of performing the pipelined convergecast through a BFS tree of the graph, our protocol performs many pipelined convergecasts *in parallel* through all trees of an appropriate *neighborhood cover* (the latter is a collection of trees that satisfy certain properties). Some properties of the trees of the cover are crucial for our analysis. We next discuss some of them. First, all the trees are rather shallow (the maximum depth is bounded from above in terms of the *MST*-radius), and, thus, potentially, the convergecasts can be conducted through them efficiently. Second, every edge of the graph belongs to a relatively small number of trees, and thus, no edge is ever utilized in parallel by too many convergecasts, ensuring that the network experiences a limited number of congestions. Third, every vertex v participates in convergecasts in each of the trees of the cover that contain v, and combines the vth local perspective of these convergecasts in a certain way. The result of this process turns out to be essentially equivalent to the result that the vertex v would obtain from participation in a single convergecast through a BFS tree of the entire graph. Essentially, the last property is an instantiation of a general paradigm that many *global* computations can be replaced by somewhat more involved *local* ones (see [Pel00] for a comprehensive discussion of this paradigm).

The main difficulty in implementing the above scheme is, however, that before conducting the pipelined convergecasts over the trees of the neighborhood cover, this cover needs to be constructed. Neighborhood covers play an important role in the design of distributed protocols, and were subject to extensive research [AP90a, AP90b, AP92, ABCP93, ABCP96]. However, today's best-known distributed protocols for constructing neighborhood covers from scratch in the $\mathcal{CONGEST}$ model that apply for the entire range of the parameters have super-linear running time [ABCP93, AP90b]. In this paper we present a drastically more efficient randomized distributed protocol for constructing neighborhood covers, and in this section this protocol is used for constructing the *MST*.

3.2 Protocol for Constructing an *MST* We first introduce the following definition. A *k-MST forest* $\mathcal{F}$ of a graph $(G = (V, E), \omega)$ is a collection of vertex-disjoint trees that satisfy

1. $\bigcup_{T\in\mathcal{F}} V(T) = V$, $\bigcup_{T\in\mathcal{F}} E(T) \subseteq E$.
2. $|V(T)| = \Omega(k)$, $depth(T) = O(k)$.
3. Each tree $T \in \mathcal{F}$ is a fragment (that is, a connected subtree) of the *MST* of the graph (G, ω).

The notion of a *k-MST* forest is related to the notion of a (σ, ρ) spanning forest of [KP95]. Trees $T \in \mathcal{F}$ of (σ, ρ) spanning forest $\mathcal{F}$ have to satisfy properties (1) and (2) with $|V(T)| \geq \sigma$ and $depth(T) \leq \rho$, but may not satisfy property (3). It was demonstrated in [KP95] that a *k-MST* forest of an n-vertex graph G can be constructed in $O(k \cdot \log^* n)$ rounds of distributed computation in the $\mathcal{CONGEST}(B)$ model for $B = \Omega(\log n)$.

Fix k to be a positive integer parameter. The first step of our protocol is to construct the *k-MST* forest.

For the following discussion we need the notion of *matroid*. Suppose we are given a universe $\mathcal{A}$ of elements, and a collection $\mathcal{S}$ of subsets that is closed under inclusion, i.e., that $A \in \mathcal{S}$ and $B \subseteq A$ implies that $B \in \mathcal{S}$. The sets of $\mathcal{S}$ are called the *independent sets* of the matroid, and a set $A \in \mathcal{S}$ is called *maximal independent set* if for every element $e \in \mathcal{A} \setminus A$, the set $A \cup \{e\} \notin \mathcal{S}$. The collection (along with the universe) is called a *matroid* if for any two sets $A, B \in \mathcal{S}$ such that $|B| = |A| + 1$, there exists an element $e \in B$ such that $A \cup \{e\} \in \mathcal{S}$. (This property is usually called *replacement* property.)

One of the properties of matroids (that follow from the definition) is that all maximal independent sets are of the same size, and this size is called the *rank* of the matroid, denoted $rank(\mathcal{A}, \mathcal{S})$.

Given a *k-MST* forest $\mathcal{F}$, consider the collection $\hat{E}$ of inter-fragment edges, i.e., edges $(u, w) \in E$ such that $u \in V(T)$, $w \in V(T')$ for some pair of distinct fragments $T, T' \in \mathcal{F}$. Observe that the set of inter-fragment edges forms a multigraph with fragments serving as vertices, and inter-fragment edges serving as the edges of the multigraph. Denote this multigraph $\hat{G} = (\hat{V}, \hat{E})$.

Observe that the collection of acyclic subsets of the edge set of a multigraph forms a matroid of rank $|\hat{V}| - 1$ with universe $\mathcal{A} = \hat{E}$. Observe also that we are given a non-negative weight function $\omega : \hat{E} = \mathcal{A} \rightarrow R^+$, and we are interested in computing a maximal independent set (henceforth, MIS) A of this matroid of *minimum weight*, $\sum_{e \in A} \omega(e)$. Such independent set is precisely the set of edges that together with the edges of the forest $\mathcal{F}$ form the MST of the graph G. Suppose also that the elements of the universe of this matroid are initially distributed with possible duplications among the vertices of a tree τ.

In [GKP93] a distributed protocol was devised for computing the MIS with minimum weight of a matroid $(\mathcal{A}, \mathcal{S})$ in this scenario. Their protocol works in the $\mathcal{CONGEST}(\log n)$ model, and its worst case running time is $O(depth(\tau) + rank(\mathcal{A}, \mathcal{S}))$. We generalize their protocol to $\mathcal{CONGEST}(B)$ model, and show that its running time in this more general model is $O(depth(\tau) + \frac{rank(\mathcal{A},\mathcal{S}) \cdot \log n}{B})$. The proof of this statement is omitted from this extended abstract. Our generalized protocol for this task will be referred henceforth Protocol *Pipeline*. It follows that under the assumption that the inter-fragment edges of $\hat{G}$ are distributed among the vertices of τ, the MST for $\hat{G}$ can be constructed in $O(depth(\tau) + \frac{|\hat{V}|}{B} \cdot \log n)$ rounds.

Throughout the rest of the section we assume that all the vertices accept as input the maximum MST-radius μ, i.e., for every vertex $v \in V$, $\hat{\mu}_v = \mu$. The generalization to the scenario when these upper bounds are allowed to disagree can be found in Section 4. To construct an MST given a k-MST forest, we use Protocol *Pipeline* in parallel on many auxilary trees.

The collection $\mathcal{C}$ of auxilary trees that is required for our protocol has to satisfy the following properties with $W = \mu(G, \omega)$ and $\kappa = O(\log n)$.
1. For each tree $\tau \in \mathcal{C}$, $depth(\tau) = O(W \cdot \kappa)$.
2. Each vertex $v \in V$ appears in at most $\tilde{O}(\kappa \cdot n^{1/\kappa})$ different trees $\tau \in \mathcal{C}$.
3. For each vertex $v \in V$ there exists a tree $\tau \in \mathcal{C}$ that contains the entire W-neighborhood of the vertex v, i.e., $\hat{\Gamma}_W(v) \subseteq V(\tau)$.
Such a collection of trees is called *sparse* (κ, W)-*neighborhood cover* (henceforth, (κ, W)-*cover*) of the graph G. For a cover $\mathcal{C}$ and a vertex $v \in V$, let the *cover-degree* of the vertex v in the cover $\mathcal{C}$, denoted $covdeg_{\mathcal{C}}(v)$, be the number of clusters of the cover $\mathcal{C}$ that contain the vertex v. The *(maximum) cover-degree* of the cover $\mathcal{C}$, denoted $covdeg(\mathcal{C})$, is $\max\{covdeg_{\mathcal{C}}(v) \mid v \in V\}$. We devise a new randomized protocol for constructing (κ, W)-neighborhood covers with maximal cover-degree $O(\kappa \cdot n^{1/\kappa} \cdot \log n)$ with running time $O(\kappa^2 \cdot \log n \cdot n^{1/\kappa} \cdot W)$. Our construction is based on a parallel algorithm for constructing *pairwise covers* due to [Coh93], and adapts some techniques from [ABCP96]. The formal description of our construction of covers is omitted from this extended abstract.

The first step of our protocol for constructing the MST given a k-MST forest is to construct a $(\log n, \mu)$-cover of the graph. As was already mentioned, this requires $O(\log^3 n \cdot \mu)$ rounds.

Let $\mathcal{C}$ denote the constructed $(\log n, \mu)$-cover. Consider some tree $\tau \in \mathcal{C}$. Its vertex set $V(\tau)$ induces a subgraph of G, $G^\tau = G(V(\tau))$. Let $\mathcal{F}$ be the k-MST forest. Let $\mathcal{F}^\tau = F|_{V(\tau)}$ be the restriction of the forest $\mathcal{F}$ to the vertex set $V(\tau)$. Observe that $\mathcal{F}^\tau$ is not necessarily a k-MST forest of G^τ, because each fragment (tree) T of $\mathcal{F}$ may form several disconnected fragments (trees) $T_1^\tau, T_2^\tau, \ldots, T_p^\tau$ for some $p = 1, 2, \ldots$. For each fragment $T \in \mathcal{F}$, let T^τ denote the forest with vertex set $V(T) \cap V(\tau)$ and edge set $\{e = (u, w) \in E(T) \mid u, w \in V(\tau)\}$.

Consider the multigraph $\hat{G}^\tau$ that is obtained out of G^τ by contracting each (not necessarily connected) fragment T^τ (such that $T \in \mathcal{F}$) to a supervertex. For each inter-fragment edge (u, w) with $u \in V(T_1^\tau)$, $w \in V(T_2^\tau)$, there is an edge labeled (u, w) between T_1^τ and T_2^τ in $\hat{G}^\tau$.

Next, Protocol *Pipeline* is invoked on each tree τ of the cover $\mathcal{C}$ to compute the MST of the multigraph $\hat{G}^\tau$. We refer to this stage of the computation as Protocol *Pipe_MST*. Observe that for each tree $\tau \in \mathcal{C}$, the acyclic sets of edges of the multigraph $\hat{G}^\tau$ form a matroid of rank equal to the number of supervertices of the multigraph $\hat{G}^\tau$ minus 1. As each fragment $T \in \mathcal{F}$ yields at most one (not necessarily connected) fragment $T^\tau = (V(\tau) \cap V(T), \{e = (u, w) \in E(T) \mid u, w \in V(\tau)\}$, it follows that the number of supervertices of $\hat{G}^\tau$ is at most the number of fragments of the k-MST forest $\mathcal{F}$, that is, $O(n/k)$. Hence, the rank of the matroid is $O(n/k)$, and, therefore, Protocol *Pipeline* would compute the MST of $\hat{G}^\tau$ in $O(depth(\tau) + \frac{n \cdot \log n}{k \cdot B}) = O(\mu \cdot \log n + \frac{n \cdot \log n}{k \cdot B})$ rounds, if this computation would proceed uninterruptedly.

However, each vertex may participate in $O(\log^2 n)$ trees $\tau \in \mathcal{C}$, and, therefore, on each round $O(\log^2 n)$ different executions of Protocol *Pipeline* may try to utilize the same edge. Hence, the running time of Protocol *Pipe_MST* is at most by a factor $O(\log^2 n)$ greater than it would be for a single execution of Protocol *Pipeline*, that is, $O(\mu \cdot \log^3 n + \frac{n \cdot \log^3 n}{k \cdot B})$.

The last step of our protocol is to form the MST of the graph G out of the k-MST forest $\mathcal{F}$, and the MSTs of the multigraphs $\hat{G}^\tau$, for each tree $\tau \in \mathcal{C}$.

This is done locally by each vertex v in the following way. For each edge $e = (u,w)$ that is incident to the vertex v, the edge e is taken into the MST if it belongs to one of the following two sets:
(1) The set of edges of the k-MST forest $\mathcal{F}$.
(2) $\{e = (v,w) \mid \forall\tau \in \mathcal{C} \text{ s.t. } v,w \in V(\tau), e \in MST(\hat{G}^{\tau})\}$.
The first condition means that the constructed MST contains the edge set of the k-MST forest $\mathcal{F}$. The second condition means that an inter-fragment edge e belongs to the constructed MST if it belongs to the MST of each multigraph G^{τ} such that the vertex set $V(\tau)$ contains both its endpoints.

We refer to the above protocol for constructing MST given a k-MST forest $\mathcal{F}$, *Cover_MST*.

LEMMA 3.1. *If $e = (u,w)$ is an inter-fragment edge that belongs to the MST of G, then it belongs to the MST of each multigraph $\hat{G}^{\tau}$ such that both its endpoints u and w belong to the vertex set $V(\tau)$.*

A cover $\mathcal{C}'$ is said to *coarsen* the cover $\mathcal{C}$, if for every cluster $C \in \mathcal{C}$, there exists a cluster $C' \in \mathcal{C}'$ such that $C \subseteq C'$. Unlike the previous lemma that does not depend on the properties of the constructed cover $\mathcal{C}$, the next lemma heavily exploits the assumption that the cover $\mathcal{C}$ coarsens a (κ,μ)-neighborhood cover.

LEMMA 3.2. *Let $e = (u,w)$ be an inter-fragment edge that does not belong to the MST of G. Let $\mathcal{C}$ be a cover that coarsens a (κ,μ)-neighborhood cover. Then there exists a tree $\tau \in \mathcal{C}$ with $u,w \in V(\tau)$, such that e does not belong to the MST of the multigraph $\hat{G}^{\tau}$.*

Proof. Consider an edge $e = (u,w)$ that does not belong to the MST of G. Observe that $VxEdgElimRad(u,e) \le \mu_u(G,\omega) \le \mu$. By definition of $VxEdgElimRad(u,e)$, there exists a cycle $C_0 \in \mathcal{Z}(e)$ such that $VxEdgRad_u(C_0) = VxEdgElimRad(u,e) \le \mu$. As $C_0 \in \mathcal{Z}(e)$, the edge e belongs to $E(C_0)$, and the endpoints u and w of e belong to $V(C_0)$. By definition of neighborhood cover, and since the cover $\mathcal{C}$ coarsens a (κ,μ)-neighborhood cover, it follows that there exists a tree $\tau \in \mathcal{C}$ such that the entire μ-neighborhood of the vertex u is contained in $V(\tau)$. It follows that the vertex set of the cycle C_0 is contained in $V(\tau)$ as well, and, in particular, $u,w \in V(\tau)$. The cycle C_0 induces a cycle C_0^{τ} in the multigraph $\hat{G}^{\tau}$, and as the edge e is an inter-fragment edge with both endpoints in $V(\tau)$, it is the heaviest edge of the cycle C_0^{τ}. It follows that the edge e does not belong to the MST of the multigraph $\hat{G}^{\tau}$, proving the lemma. ▮

COROLLARY 3.1. *Given a k-MST forest $\mathcal{F}$ of an n-vertex graph (G,ω), Protocol Cover_MST constructs the MST of (G,ω) with probability $1 - O(1/poly(n))$ in $O(\mu \cdot \log^3 n + \frac{n}{k \cdot B} \cdot \log^3 n)$ rounds.*

Proof. Suppose that the neighborhood cover was constructed by Protocol *Cover* correctly. This happens with probability $1 - O(1/poly(n))$.

Consider an edge e of the MST of G. If it is an inter-fragment edge (with respect to the forest $\mathcal{F}$), then by Lemma 3.1 and by condition (2) of Protocol *Cover_MST*, it belongs to the tree τ_0 that was constructed by Protocol *Cover_MST*.

If the edge $e = (u,w)$ is not an inter-fragment edge, then there exists a fragment $T \in \mathcal{F}$ such that $u,w \in V(T)$. By definition of the k-MST forest, T is a connected subtree of the MST, and, therefore, it spans all the vertices of $V(T)$. Hence, for any edge $e' = (v,z)$ of the MST with both endpoints v and z in $V(T)$, the edge e' belongs to the edge set of the fragment. Hence, in particular, $e \in E(T)$, and, therefore, it belongs to the tree τ_0 that was constructed by the protocol.

Consider an edge $e = (u,w)$ that does not belong to the MST. If the edge is inter-fragment with respect to the forest $\mathcal{F}$, then by Lemma 3.2 and condition (2), the edge e does not belong to τ_0. Consider the case that both endpoints u and w of the edge e belong to the same fragment of $\mathcal{F}$. As the edge does not belong to the MST, it does not belong to the k-MST forest $\mathcal{F}$. Hence, it was not inserted into the tree τ_0.

It follows that the tree τ_0 is precisely the MST of the graph.

Regarding the running time, observe that constructing the neighborhood cover requires $O(\mu \cdot \log^3 n)$ rounds. Recall that running Protocol *Pipeline* in parallel on all the trees τ of the neighborhood cover $\mathcal{C}$ requires $O(\mu \cdot \log^3 n + \frac{n}{k \cdot B} \cdot \log^3 n)$ rounds. The corollary follows. ▮

Set $k = \sqrt{\frac{n}{B}} \cdot \frac{\log^{3/2} n}{\sqrt{\log^* n}}$. Let Protocol *Fast_MST* denote the protocol that computes k-MST forest for $k = k(n,B)$ as above, using the protocol due to [GKP93], and then invokes Protocol *Cover_MST*. Recall that computing the k-MST forest requires $O(k \cdot \log^* n)$ rounds. We conclude

THEOREM 3.1. *Protocol Fast_MST computes the MST of an n-vertex graph $(G = (V,E),\omega)$ from scratch assuming that every vertex v accepts as input the maximum MST-radius $\mu(G,\omega)$ of the graph in $O(\mu \cdot \log^3 n + \sqrt{n/B} \cdot \log n \cdot \sqrt{\log^* n})$ rounds of distributed computation with success probability $1 - O(1/poly(n))$.*

4 Extensions of Protocol *Fast_MST*

In this section we generalize Protocol *Fast_MST* to work in the scenario when each vertex $v \in V$ accepts as input only an upper bound $\hat{\mu}_v$ of the maximum *MST*-radius μ, and not the maximum *MST*-radius itself. (For different vertices $u, w \in V$, the upper bounds $\hat{\mu}_u$, $\hat{\mu}_w$ may disagree.) Let $\hat{\mu} = \max\{\hat{\mu}_v \mid v \in V\}$. We will argue that in this scenario the *MST* of the input graph can be constructed in $\tilde{O}(\hat{\mu} + \sqrt{\frac{n}{B}})$ rounds.

We start with describing the generalized protocol, to which we refer as Protocol *Gen_Fast*. Like in Protocol *Fast_MST*, the first part of Protocol *Gen_Fast* is to construct the *k-MST* forest. The second part of Protocol *Gen_Fast* is an invocation of Protocol *Gen_Cov_MST*, that generalizes Protocol *Cover_MST*. Protocol *Gen_Cov_MST* proceeds as follows. It starts with initializing the cover $\mathcal{C}$ to be an empty set, i.e., each vertex v initializes its local set $\mathcal{C}(v)$ of clusters of the cover $\mathcal{C}$ that contain v as an empty set. From now on these sets will only grow, and once a cluster gets into one of these sets, it is never removed from it. Also, initially all the vertices marked as not terminated.

Protocol *Gen_Cov_MST* proceeds in phases. On phase $j = 1, 2, \ldots$, all the vertices v that are not terminated, and such that $\hat{\mu}_v > 2^{j-1}$ invoke Protocol *Cover* with $\kappa = \log n$ (the κ-parameter will stay unchanged throughout the protocol), and $W = W_j = 2^j$. Let $\mathcal{C}^j$ denote the constructed cover. Each vertex v as above adds all the clusters of $\mathcal{C}^j$ that contain it into its local set $\mathcal{C}(v)$. Note that in this stage of the protocol, the vertices maintain a cover $\bigcup_{\ell=1}^{j} \mathcal{C}^\ell$. We denote the latter cover as $\bar{\mathcal{C}}^j$.

Next, all the vertices v that participated in the execution of Protocol *Cover* on this phase invoke a modified variant of Protocol *Pipe_MST* on the cover $\bar{\mathcal{C}}^j$. Specifically, recall that during Protocol *Pipe_MST* (see Section 3) the edges of the graph are pipelined through the different trees of the cover $\bar{\mathcal{C}}^j$. Recall also that in Protocol *Fast_MST* the sets of edges that are pipelined through the trees of the cover form a matroid of rank $O(n/k)$. The latter is not necessarily true in Protocol *Gen_Fast*, because the W-parameter of the constructed cover $\bar{\mathcal{C}}^j$ may be not large enough to ensure the acyclicity of the pipelined sets of edges. Therefore, using Protocol *Pipe_MST* as is would result in an untolerably high running time.

Instead, this protocol is run with an additional threshold parameter $t = O(n/k)$, and whenever a vertex needs to pipeline t' items with $t' > t$ items, it discards some arbitrary $t' - t$ of them. We refer to this modified version of Protocol *Pipe_MST* as Protocol *Gen_Pipe*. Observe that whenever Protocol *Gen_Pipe* runs on a matroid of rank $O(n/k)$, it never needs to apply the threshold condition, and, therefore, its execution in this case is identical to the one of Protocol *Pipe_MST*. In other words, whenever the W-parameter of the constructed cover is large enough, Protocol *Gen_Pipe* behaves identically to Protocol *Pipe_MST*. Also, in either case the running time of Protocol *Gen_Pipe* is at most $O(depth(\tau) + \frac{n \cdot \log n}{k \cdot B})$, where τ is the tree on which it was invoked (that is, the same upper bound as for Protocol *Pipe_MST*).

If throughout the execution of phase j the vertex v was informed that one of the vertices in its vicinity is already terminated, then v will terminate at the end of the current phase. Also, if the index j of the current phase satisfies $2 \cdot \hat{\mu}_v > 2^j \geq \hat{\mu}_v$, then the vertex v will also terminate at the end of the current phase. In both cases v will produce its output prior to the termination. In the former case (when a terminated vertex was discovered in the vicinity of v) the output of v will be the output of $(j-1)$st phase restricted to v, and in the latter case the output of v will be the output of jth phase restricted to v. Also, on the last round of each phase the vertices update their neighbors of whether they decide to terminate. However, the vertices that are informed on this stage that one of their neighbors terminate, will terminate only at the end of the next phase; otherwise, either the process of informing the neighbors could require $O(\Lambda(G))$ rounds, which exceeds our time limits, or some vertices would have terminated neighbors that they would not know about.

Finally, there are two possible ways for a vertex v to be informed throughout phase ℓ that one of the vertices in its vicinity is terminated is as follows. First, throughout the execution of Protocol *Cover* the vertex v may turn out to be explored by the same BFS exploration as some terminated vertex w, and then all the vertices of this exploration are informed that w is terminated (incurring only a constant overhead in running time). Second, when Protocol *Gen_Pipe* performs pipelined broadcasts and convergecasts through the trees of the cover $\bar{\mathcal{C}}^\ell$, it may discover a terminated vertex in one of the trees of the cover $\bar{\mathcal{C}}^{\ell-1} \subseteq \bar{\mathcal{C}}^\ell$ (since a terminated vertex in a cover $\mathcal{C}^\ell = \bar{\mathcal{C}}^\ell \setminus \bar{\mathcal{C}}^{\ell-1}$ would be discovered through the execution of Protocol *Cover*), and one of these trees may contain v. In this case, again, all the vertices of the tree in which a terminated vertex was encountered through the execution of Protocol *Gen_Pipe* are informed, and the overhead in the running time is again at most constant.

Note that Protocol *Gen_Pipe* performs pipelined broadcasts and convergecasts through the trees of this constructed cover, and, so, if the all the vertices of one these trees were not terminated throughout the construction of the cover, they will still be not terminated throughout the execution of Protocol *Gen_Pipe*.

This completes the description of the modified protocol *Gen_Fast*. Its analysis is omitted from this extended abstract.

Acknowledgements

The author is grateful to Oded Regev for very stimulating discussions about the every-case optimality. The author is also grateful to Avi Wigderson for suggesting a simpler definition of the every-case complexity measure, and to Alexander Razborov for suggesting the notion of will-maintaining algorithms. Finally, the author wishes to thank Oded Goldreich, David Peleg, Guy Kortsarz, and Yehuda Hassin for helpful discussions.

References

[Awe85] B. Awerbuch, Complexity of network synchronization. In *J. ACM*, 4:804-823, 1985.

[Awe87] B. Awerbuch, Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems. In *Proc. 19th ACM Symp. on Theory of Computing*, pp. 230-240, May 1987.

[Awe89] B. Awerbuch, Distributed shortest paths algorithms. In *Proc. 21st ACM Symp. on Theory of Computing*, pp. 230-240, May 1989.

[ABCP93] B. Awerbuch, B. Berger, L. Cowen and D. Peleg, Near-linear cost sequential and distributed constructions of sparse neighborhood covers. In *Proc. Proc. 34th IEEE Symp. on Foundations of Computer Science*, pp. 638-647, 1993.

[ABCP96] B. Awerbuch, B. Berger, L. Cowen and D. Peleg, Fast Distributed Network Decompositions and Covers. Journal of Parallel and Distributed Computing, 39:2: 105-114. 1996.

[AG87] B. Awerbuch and G. Gallager, A new distributed algorithm to find breadth first search trees. *IEEE Trans. on Information Theory*, IT-33(3):315-322, 1987.

[AGLP89] B. Awerbuch, A. Goldberg, M. Luby and S. Plotkin, Network decomposition and locality in distributed computation. in *Proc. 30th IEEE Symp. on Foundations of Computer Science*, pp. 364-369, 1989.

[AP90a] B. Awerbuch and D. Peleg, Network synchronization with polylogarithmic overhead. In *31st IEEE Symp. on Foundations of Computer Science*, pp. 514-522, Oct. 1990.

[AP90b] B. Awerbuch and D. Peleg, Efficient distributed construction of sparse covers. Technical Report CS90-17, The Weizmann Institute of Science, Rehovot, Israel, July 1990.

[AP92] B. Awerbuch and D. Peleg, Routing with polynomial communication-space tradeoff. *SIAM J. Discrete Math.*, 5:151-162, 1992.

[AS92] N. Alon and J. H. Spencer, The Probabilistic Method, Wiley, 1992, p.239.

[CT85] F. Chin and H.F. Ting, An almost linear time and $O(n \log n + e)$ messages distributed algorithm for minimum-weight spanning trees, in *Proc. 26th IEEE Symp. on Foundations of Computer Science*, Los Alamitos, CA, 1985, pp.257-266.

[Coh93] E. Cohen, Fast Algorithms for constructing t-spanners and paths of stretch t, in *Proc. 34th IEEE Symp. on Foundations of Computer Science*, IEEE, Piscataway, NJ, 1993, 648-658.

[DM90] C. Dwork and Y. Moses, Knowledge and common knowledge in a Byzantine environment: Crash failures, in *Information and Computation*, 88(2):156-186, Oct. 1990.

[Elk01] M. L. Elkin, Computing Almost Shortest Paths, in *Proc. 20th ACM Symp. on Principles of Distributed Computing*, pp. 53-62, Newport, Rhode Island, August, 2001.

[Elk03] M. L. Elkin, Unconditional Lower Bounds on the Time-Approximation Tradeoff for the Distributed Minimum Spanning Tree Problem, manuscript, 2003.

[FLN01] R. Fagin, A. Lotem, and M. Naor, Optimal Aggregation Algorithms for Middleware, in *Proc. 20th ACM Symp. on Principles of Database Systems*, 2001, pp. 102-113.

[Gaf85] E. Gafni, Improvements in the time complexity of two message-optimal election algorithms, in *Proc. 4th Symp. on Principles of Distributed Computing*, ACM, New York, 1985, pp. 175-185.

[GHS83] R.G. Gallager, P.A. Humblet and P.M. Spira, A distributed algorithm for minimum-weight spanning trees, *ACM Trans. on Programming Lang. and Syst.*, 5:66-77, 1983.

[GKP93] J.A. Garay, S. Kutten and D. Peleg, A sublinear time distributed algorithm for minimum-weight spanning trees, in *Proc. of 34th IEEE Symp. on Foundations of Computer Science*, Palo Alto, CA, IEEE Computer Society Press, Los Alamitos, CA, 1993, pp. 659-668.

[KP95] S. Kutten and D. Peleg, Fast distributed construction of k-dominating sets and applications, in *Proc. of 14th ACM Symp. on Principles of Distributed Computing*, Ottawa, Canada, August, 1995, pp. 20-27.

[Lev72] L. A. Levin, Universal Search Problems, in *Problemy Peredaci Informacii 9*, pp. 115–116, 1973. Translated in *problems of Information Transmission 9*, pp. 265–266.

[Lin92] N. Linial, Locality in distributed graph algorithms. *SIAM J. Comput.*, 21:193-201,1992.

[LS91] N. Linial and M. Saks, Decomposing graphs into regions of small diameter, In *Proc.of the 2nd Symp. on Discrete Algorithms*, ACM, pp.320-330, 1991.

[Pel00] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*, SIAM, Philadelphia, PA, 2000.

[Pel00a] "———", pp.273-275.

[PR99] D. Peleg and V. Rubinovich, A near-tight lower bound on the time complexity of distributed MST construction, In *Proc. 40th IEEE Symp. on Foundations of Computer Science*, pp.253-261, 1999.