Distributed small connected spanning subgraph

Breaking the diameter bound

Distributed Small Connected Spanning Subgraph: Breaking the Diameter Bound

L. Shankar Ram, Elias Vicari

ETH Zurich,
Institute of Theoretical Computer Science,
8092 Zurich, Switzerland.
E-mail: {lshankar,vicariel}@inf.ethz.ch.

ETH Technical Report - 530

Abstract

In any ad hoc or a wireless sensor network, the question of designing efficient distributed algorithms for connectivity problems is important. We study one such problem here. The Small Connected Spanning Subgraph Problem (SCSSP) is the following: Given a connected network $G = (V, E)$ and a parameter $|V| - 1 \leq k \leq |E|$ find a connected subgraph $S = (V, E')$ where $E' \subseteq E$ such that $|E'| \leq k$. If $k = |V| - 1$ then the subgraph is a spanning tree. It is known that any distributed algorithm needs $d(G)$ rounds to compute a spanning tree where $d(G)$ is the diameter of G. In this paper, we design a simple distributed algorithm that finds a connected spanning subgraph with $(1+\varepsilon)n$ edges in $\min\{d(G), \mathcal{O}(\log n)\}$ rounds, for any $\varepsilon > 0$. We reduce the message complexity of this algorithm and provide another algorithm which also finds a connected spanning subgraph with $(1 + \varepsilon)n$ edges but runs in $\min\{d(G), \mathcal{O}(\log n \cdot \log\log n \cdot \log^{\star} n)\}$ rounds. Moreover, we show that any synchronous distributed algorithm running for $o(\log n)$ rounds computes a connected spanning subgraph with $\omega(n)$ edges. Our results make use of a famous theorem from extremal graph theory and hold for arbitrary graphs.

1 Introduction

The design of distributed algorithms for problems appearing in the realm of ad hoc and sensor wireless networks has gained considerable momentum in the recent years. One of the main reasons for this is that the distributed

paradigm lies in the bridge connecting theory and practice. Several problems from theory have been considered in the distributed setting and practical implementations have been successful [KMW05].

An ad hoc network is comprised of autonomous nodes that communicate with each other via radio equipment without any apriori information on the structure of the whole network. In other words, if two nodes are not within direct transmission range then they communicate through other intermediate nodes (relays). Since there is no clear structure in such a network, as the name suggests, almost every solution relies on a dominating set based clustering approach [AWF02, KMW04, WL99]. Clustering facilitates effective routing [AWF02, WL99] and efficient network initialization [KMW04, MvRW06].

Our distributed model is based on message passing with the further assumption that all the algorithms are synchronous i.e., it takes the same time for a message to transfer from one endpoint to another of an edge, for all the edges and the transmission starts simultaneously at all the nodes. The local computations performed by a node in between the rounds are not taken into consideration for measuring the time complexity. Every node communicates with its neighbours using messages. In one *round*, any node can broadcast messages to some or all of its neigbours. Thus an important parameter in determining the complexity of a distributed algorithm is the time complexity (number of rounds). Another interesting parameter is the message complexity or the size of a message between two nodes. We state two assumptions that we make in our model. These assumptions, in no way, reduce the complexity of a given problem but only help to simplify the description and the analysis.

In the distributed setting of sensor networks, a meaningful algorithm is one which runs in $o(d(G))$ rounds of synchronous computation. Here $d(G)$ is the diameter or the length of a longest shortest path in G. This is because, in $d(G)$ rounds every node of G knows the whole graph and hence in particular, any substructure of G can be computed.

Computing spanning trees is one of the fundamental problems in the theory of networks. Naturally, in the distributed setting it assumes even more importance. Several asymptotic near-optimal distributed protocols are known for the problem [KP98, Elk04a, Elk04b]. For example, Kutten and Peleg [KP98] design a protocol with a running time of $\mathcal{O}\left(d(G) + \sqrt{n} \cdot \log^* n\right)$. Note that the running time required exceeds the diameter of the graph in the worst case and therefore it is not suitable for highly dynamic systems as wireless sensor networks.

In this paper, we consider the following problem (Small Connected Spanning Subgraph Problem or SCSSP): Given a connected network $G = (V, E)$

and a parameter $|V| - 1 \leq k \leq |E|$ find a connected subgraph $S = (V, E')$ where $E' \subseteq E$ such that $|E'| \leq k$. Clearly when $k = |V| - 1$, the subgraph of interest is a spanning tree.

In the central setting (non-distributed) the focus was directed towards minimum-weight spanning trees and there efficient algorithms do exist (see for example [CLR90]). The $\mathcal{O}(m \log m)$-textbook-algorithms were successively improved up to an algorithm with a running time of $O(m \log \beta(m, n))$ due to [GGST86] where $\beta(m, n)$ is the minimum integer i such that the value obtained by iterated logarithm on n is less than $\frac{m}{n}$ with $n = |V|, m = |E|$. Fourteen years later, the running time has been further improved by Chazelle [Cha00] up to the current optimal algorithm due to Pettie and Ramachandran [PR00], who show a deterministic comparison-based algorithm that finds a solution in $\mathcal{O}(\mathcal{T}^*(m, n))$ time, where $\mathcal{T}^*(m, n)$ is the number of edge-weight comparisons needed to determine the solution. Surprisingly, in the distributed setting, a spanning tree cannot be found in $o(d(G))$ rounds where $d(G)$ is the diameter of the graph G. One can deduce this result from the impossibility of electing a leader in a ring (a cycle of n nodes) within n rounds of synchronous computation (for example see [Awe87]). Therefore, we consider the problem of finding connected spanning subgraphs for other values of k.

The SCSSP problem can be motivated by the following application. In many networks, broadcasting or the ability of a node to transmit a message to all other nodes, is an important requirement. Thus, one can imagine that an efficient algorithm for the SCSSP problem can be directly used for broadcasting since minimum number of edges are used in the connected spanning subgraph. Broadcasting is important for many tasks on a network, as counting the number of nodes, spread a message to all nodes and so on. The problem can be related to graph spanners or tree spanners where the aim is to compute a spanning subgraph where distance between nodes is bounded by at most a factor (usually called the *stretch factor*) of the actual distance in the original graph (for a survey see [Soa92]).

In this paper, we design a simple distributed algorithm that finds a connected spanning subgraph with $(1 + \varepsilon)n$ edges in $\min\{d(G), \mathcal{O}(\log n)\}$ rounds for any approximation ratio $\varepsilon > 0$. We reduce the message complexity of this algorithm and provide another algorithm which also finds a connected spanning subgraph with $(1 + \varepsilon)n$ edges but runs in $\min\{d(G), \mathcal{O}(\log n \cdot \log \log n \cdot \log^\star n)\}$ rounds. Here $\log^\star n$ is the minimum integer t such that the value of the iterated logarithm on n for t steps is less than 2. Moreover, we show that any synchronous distributed algorithm running for $o(\log n)$ rounds computes a connected spanning subgraph with $\omega(n)$ edges. Our results make use of a famous theorem (Turan type) from extremal graph

theory. Up to the knowledge of the authors, the results of this paper are the first ones for this problem. The paper is organized as follows: in Section 2 we introduce the distributed model that we will be working with. In Section 3 we describe our algorithm followed by a reduction of message complexity of this algorithm in the subsequent section. Finally in Section 5 we give the lower bound for the problem.

2 The Model

An ad hoc or a sensor wireless network can be modelled as a simple graph $G = (V, E)$. The vertex-set V represents the devices of the network that can communicate directly if the corresponding nodes are connected by an edge in G. Usually one assumes G to be a *unit-disk* graph (the intersection graph of unit disks on a plane), a *quasi unit-disk graph*, or a *growth-bounded graph* (for a formal definition see [KMW05]) which is a large class containing the former two classes in the wireless sensor network community. These classes are both suitable in practice (they model the reality up to some extent) and have nice theoretical properties. However, in this work we do not require the network to be belonging to any of the aforementioned special graph classes. All our results hold for arbitrary graphs.

As mentioned in the introduction, the goal of SCSSP is to compute a connected spanning subgraph of a given graph with as few edges as possible. Every node of the network is given a unique ID (for example the IP-number in a computer network) which induces an ordering on the edges. An edge $s = xy$ (wlog $\text{ID}(x) > \text{ID}(y)$) is said to be *bigger* than an edge $t = uv$ ($\text{ID}(u) > \text{ID}(v)$), if $\text{ID}(x) > \text{ID}(u)$ or if $x = u$ and $\text{ID}(y) > \text{ID}(v)$. We require that every node knows n, the number of nodes in the graph.

We assume that the nodes are *time-aware* having access to a global clock allowing a synchronous execution of the program they run. In the literature, different type of *synchronizers* are described, which extend a result for synchronous network elements to asynchronous networks at the cost of increased time and message complexities.

In every round, each node can send a message to all of its neighbors. Although we give upper bounds on message sizes at some places, we generally assume that message size is unbounded and that there is no restriction to local computations. These assumptions are not restrictive [KMW05].

Throughout this paper we make use of some standard notation: K_n is the complete graph on n vertices, $N_v(t)$ is the set of nodes at distance at most t to v (v included), $d_H(x, y)$ is the number of edges of a shortest path between x and y according to a subgraph H of G and n is the size of the

network, i.e. the number of nodes of G. Let $d(G)$ be the diameter of the graph where

$$d(G) := \max_{x,y} d_H(x, y).$$

In Section 4 we make use of the maximal independent set problem (MIS) and the notion of a κ-ruling set. The former is the problem of finding a subset $S \subset V$ with the property that no two nodes of S are connected in G and such that no superset $S' \supseteq S$ has this property. A κ-ruling set is a subset $S \subset V$ such that every node in the graph is at a distance at most κ to a vertex in S (*ruling property*). Note that an MIS is a 1-ruling set.

3 The Distributed Algorithm

In this section we describe a distributed algorithm for the SCSSP. We improve the message complexity of this algorithm in the subsequent section. Our algorithm works as follows: we explore the $c(\varepsilon) \log n$-neighbourhood ($N_v(c(\varepsilon) \log n)$) for all vertices $v \in V$. In such a neighbourhood, every v computes locally all the cycles and removes the lexicographically biggest edge of every cycle in the increasing order of the length of the cycles. We refer to this process of removing the biggest edge of a cycle as *destroying* a cycle. Note that by our assumption of the model, all these local computations are not measured in the time complexity of the algorithm.

Now we analyze Algorithm 1 showing that it builds a connected spanning subgraph with few edges. For bounding the number of edges returned by the algorithm, we make use of theorems from *Turan's Theory* (see [Bol04]) of extremal combinatorics.

3.1 Results from Extremal Graph Theory

Turan's theory deals with the question of bounding the maximal number of edges that a graph G on n vertices can possibly have avoiding a given subgraph H. We denote this quantity by $\mathrm{ex}(n, H)$ when the underlying graph G is the complete graph K_n. An interesting theorem was proved by Mantel in 1907 which states that $\mathrm{ex}(n, K_3) = \lfloor n^2/4 \rfloor$, which was further generalized by Turan in 1941 to $\mathrm{ex}(n, K_s) = \left(1 - \frac{1}{s-1}\right) \frac{n^2}{2}$ (see [Bol04]). A celebrated result by Erdős and Stone [ES46] solves the problem meaningfully for all non-bipartite graphs H by proving

$$\mathrm{ex}(n, H) = \left(1 - \frac{1}{\chi(H) - 1}\right) \frac{n^2}{2} + o(n^2)$$

Algorithm 1: Small Spanning Subgraph

Algorithm:SPANNING_SUBGRAPH

Input: A graph $G = (V, E)$, an approximation ratio $\varepsilon > 0$

Output: A connected spanning subgraph of G with $(1+\varepsilon)n$ edges.

1 Set $c := 6/\log(1+2\varepsilon)$

2 $\forall v \in V$:

3 v deletes the biggest edge for every cycle in $N_v(c \log n)$; destroy small cycles first.

4 **return** *The remaining spanning subgraph*

where $\chi(H)$ denotes the *chromatic number* of H. Several results are known when the subgraph to be avoided belongs to other classes of graphs. If we have to avoid more than one type of subgraph, the extremal number is denoted $\mathrm{ex}(n, H_1, H_2, \cdots, H_l)$. In other words the maximal number of edges a graph G on n vertices can have without containing any of $H_1, H_2, \cdots, H_l$ as a subgraph is given by $\mathrm{ex}(n, H_1, H_2, \cdots, H_l)$.

With a method due to Moore and improved by Alon et al. [AHL02] one can prove the upper bound of the following proposition. The lower bound is due to Margulis [Mar88] using the famous Ramanujan graphs and to Lubotzky et al. [LPS88].

Proposition 1.

$$n^{1+\frac{2}{3k+3}} \leq ex(n, C_3, C_4, \ldots, C_{2k+1}) \leq \frac{1}{2}n^{1+\frac{1}{k}} + \frac{1}{2}n$$

3.2 The Algorithm

Our algorithm returns a connected spanning subgraph with as few edges as possible, suggesting of course trees or subgraphs with not many more edges than a tree, also graphs with large girth. This is our motivation to use the above proposition. From Proposition 1, we can prove a linear upper bound on the number of edges of the subgraph returned by Algorithm 1.

Theorem 1. *For every $\varepsilon > 0$ Algorithm 1 builds a connected spanning subgraph with at most $(1+\varepsilon)n$ edges in $\min\{d(G), \mathcal{O}(\log n)\}$ rounds.*

Proof. Let $H = (V, E')$ denote the subgraph returned by the algorithm. Note that since we run the algorithm on all vertices, the subgraph is spanning. We need to verify that H is connected and $|E'| = (1+\varepsilon)n$.

Connectivity: Consider a cycle present in the $\log n$-neighbourhoods of $u, v \in V$. Since the lexicographically biggest edge of this cycle is unique, one

can ensure that always this same edge is deleted by the algorithm running on u, v respectively. Since the cycles of smallest length are destroyed first, after destroying all the cycles we are still left with a connected subgraph H.

Number of edges: H does not contain any cycle of length less than $c \log n$ (c chosen as in Algorithm description) since all the cycles within the $c \log n$-neighbourhoods of the vertices have been destroyed. Thus the remaining cycles of H have to be of length greater than $c \log n$. Therefore by using Proposition 1 H cannot contain more than

$$\mathrm{ex}(n, C_3, C_4, \ldots, C_{c \log n}) \leq \frac{1}{2} n^{1+\frac{2}{c \log n - 1}} + \frac{1}{2} n \leq \frac{1}{2} n 2^{6/c} = (1+\varepsilon) n$$

edges. ♢

Note that by choosing $\varepsilon := 1/\log n$, the algorithm constructs a connected spanning subgraph with $n + o(n)$ edges in $\mathcal{O}\left(\log^2 n\right)$ rounds.

Unfortunately, the distances in the new graph are not under control: If one considers the graph constituted by cycles of length $\mathcal{O}(\log n)$, pairwise merged on a common edge, one can define the labels in such a way, that two neighboring nodes get to distance n after the process.

3.3 The Weighted Case

Consider the case where the graph is provided with a weight function $\ell : E \to \mathbb{R}$ on the edges. $\ell(e)$ could represent the cost to use the edge e during a broadcasting transmission or the latency one incurs by sending a message along e. We are interested in a weighted version of SCSSP denoted w-SCSSP. Here the goal is to find a connected spanning subgraph of minimum total weight. Unfortunately for w-SCSSP, one cannot design a fast distributed algorithm that has a provable approximation guarantee. Consider for example a cycle of length n, where every edge has weight 1 except for one which has weight $\vartheta \geq 1$. The optimal connected spanning subgraph has a weight of $n-1$ given by a minimum weight spanning tree (MST), but every algorithm working for $o(d(G))$ rounds necessarily selects all the edges of the cycle, yielding a connected spanning subgraph of total weight $n-1+\vartheta$. This can be arbitrarily bad, since ϑ can be made big.

On the other hand, if the weight function is bounded and the instance is large enough, our algorithm provides immediately a $(1+\varepsilon)$-approximation for this problem, by choosing an appropriately large c (refer to Algorithm 1). We can suitably modify the deletion rule by choosing the heaviest edge (with respect to the weight function) in the cycle, breaking ties as before.

Algorithm 2: $(3^\kappa - 1)$-ruling set

Algorithm: RULING_SET

Input: A graph $G = (V, E)$, a desired depth κ.

Output: A $3^\kappa - 1$-ruling set S.

1 $S := V$, $\overline{G} := G$
2 **Repeat** κ *times*
3 Simultaneously for all $v \in S$: send an invitation to a $\overline{G}$-neighbor.
4 Simultaneously for all $v \in S$: accept an invitation, if any.
5 Compute a MIS M on the generated subgraph.
6 $S := M$, other nodes inactive
7 $V(\overline{G}') := S$, $E(\overline{G}') := \{xy \,|\, x, y \in V, d_{\overline{G}} \leq 5\}$
8 $\overline{G} := \overline{G}'$
9 **return** S

4 Reducing Message Complexity

Algorithm 1 although being time-optimal, requires that all the nodes in the network compute $\mathcal{O}(\log n)$-neighbourhoods to destroy cycles. This generates a huge message traffic which is not desirable and possibly not supported by the network. Here, we describe a solution that overcomes this issue by requiring only $o(n)$ nodes to compute the neighbourhoods and subsequently destroy the cycles. Of course this implies that the time complexity is increased at the cost of a reduction in the number of truely active nodes.

The first phase is a preprocessing phase where a subset of size $o(n)$ of the nodes are chosen to work in the *cycle destruction* phase. At the end of this phase, the set of nodes obtained will be referred to as *supernodes*.

According to Algorithm 2, we identify a small set S of vertices with the property that every node in $V \backslash S$ has to be close to one of them. Actually we do not need a very special structure (like an independent set here), but we can make use of the broad machinery developed for this problem to achieve our goal. To this end, we modify the ruling-set algorithm by Kuhn et al [KMNW05] which builds a degree-bounded subgraph (actually the maximum degree is two) in constant time, allowing thereby the computation of a maximal independent set (MIS) in $\mathcal{O}(\log^\star n)$ time [CV86, GPS87, Lin92].

In our algorithm for computing a ruling set, after every iteration we maintain the computed MIS. This defines an auxiliary graph (on the nodes of the ruling set) in which we recurse. The nodes of this auxiliary graph are given by the nodes of the set S and we connect two nodes if they are at a distance of at most five in G. This preserves the *ruling* property and ensures

that the number of nodes keeps decreasing in every step of recursion, because a node chosen to be in S can find another element of S in its 5-neighborhood by construction.

Lemma 1. *For any $\kappa > 1$ Algorithm 2 returns a $3^\kappa - 1$-ruling set S of at most $(2/3)^\kappa n$ vertices.*

Proof. Every node accepts exactly one invitation. So, together with its invitation, at most two edges per node are selected. If a node is not involved in the edges generated, it must have a neighbour which is. This is true for any iteration of the algorithm.

Let us prove by induction on κ. For $\kappa = 1$ we have a MIS of a degree-two graph, so at least a third of the nodes become inactive. Since every node has an involved neighbour and a MIS is a 1-ruling set, we get the 2-ruling property and the claim is established for $\kappa = 1$.

Let us assume that the induction hypothesis is true for $\kappa \geq 1$. Now consider the next iteration. If at the beginning $\overline{n} \geq (2/3)^\kappa n$ nodes were still active, we loose again at least one third of those. Thus at most $2/3\overline{n} = (2/3)^{\kappa+1} n$ nodes are still active.

Let $3^\kappa - 1$ be the ruling distance of the auxiliary graph with respect to the original graph at the beginning of the next iteration. From the above reasoning, after the new iteration every node is at most at a distance 3^κ to an active node. Since we produce again a 2-ruling set of the auxiliary graph, the ruling distance of the original graph generated after this iteration is

$$3^\kappa - 1 + 2 \cdot 3^\kappa = 3^{\kappa+1} - 1$$

which completes the proof. ♢

Note that Algorithm 2 runs in $\min\{d(G),\ \mathcal{O}(\log n \cdot \log\log n \cdot \log^\star n)\}$ rounds.

Theorem 2. *Algorithm 3 produces a connected spanning subgraph with $(1+\varepsilon)n$ edges in $\min\{d(G), \mathcal{O}(\textit{polylog } n)\}$ rounds where at most $o(n)$ supernodes destroy the cycles of G.*

Proof. Choosing $\kappa = \log\log n$, Algorithm 2 produces a $(c\log_3 n - 1)$-ruling set with $o(n)$ supernodes. If every supernode destroys cycles in its $2c \cdot \log_3 n$-neighborhood then there is no cycle of length at most $c \log n$. Applying the same argument as in the proof of Theorem 1 proves the claim. ♢

Algorithm 3: Small Spanning Subgraph with few supernodes

Algorithm:SPANNING_SUBGRAPH_WITH_SUPERNODES

Input: A graph $G = (V, E)$, an approximation ration $\varepsilon > 0$.

Output: A connected spanning subgraph of G with $(1 + \varepsilon)n$ edges.

1 Set $c := 6/\log(1 + 2\varepsilon)$

2 Apply RULING_SET on G with $\kappa := \log c \log n$

3 $\forall v \in S$:

4 v deletes the biggest edge for every cycle in $N_v(2c \log_3 n)$; destroy small cycles first.

5 **return** *Remaining edges*

5 Lower Bound

A well-known result states that it is impossible to elect a leader in a ring (a cycle on n nodes) in less than $\mathcal{O}(n)$ rounds of synchronous computation. This implies that building a spanning tree on a ring network also takes $d(C_n)$ time where $d(C_n)$ is the diameter of the ring. This is because, a better algorithm for the spanning tree could be used to solve trivially the leader election problem with an additional round.

In this section, we show a lower bound on the number of edges of the connected spanning graph produced by a distributed algorithm running for $o(\log n)$ rounds on the SCSSP problem.

Theorem 3. *Every distributed synchronous algorithm running for $o(\log n)$ rounds with the aim of building a connected spanning subgraph picks at least $\omega(n)$ edges in the worst case.*

Proof. Let us consider such an algorithm $\mathcal{A}$. Since $\mathcal{A}$ runs for $o(\log n)$ rounds, every node explores at most its $o(\log n)$-neighbourhood. Thus every node has to base its decision only according its $o(\log n)$-neighborhood and it cannot therefore prevent cycles of length at least $\Omega(\log n)$.

Let $G = (V, E)$ be an extremal graph where the length of a shortest cycle is $\Omega(\log n)$ and $|E|$ is maximum among all such graphs. If this graph is given as input to $\mathcal{A}$ then every node needs to choose all edges in its $o(\log n)$-neighborhood, in order to ensure connectivity. But this means that all the edges of the graph have to be output. Using the lower bound given in Proposition 1, this yields

$$\mathrm{ex}(n, C_3, C_4, \ldots, C_{o(\log n)}) \geq n^{1+\frac{2}{3/2(o(\log n)+1)}} = n \cdot 2^{\frac{\log n}{o(\log n)}} = \omega(n)$$

edges. ♢

Acknowledgements

We are grateful to Roger Wattenhofer for suggesting the problem.

References

[AHL02] N. Alon, S. Hoory, and N. Linial. The Moore bound for irregular graphs. *Graphs Combinatorics*, 18(1):53–57, 2002.

[Awe87] Baruch Awerbuch. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems (detailed summary). In *Proc. 19th Annual ACM Symp. on Theory of Computing*, pages 230–240, 1987.

[AWF02] K. Alzoubi, P.-J. Wan, and O. Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *MOBIHOC '02: Proceedings of the third ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 157–164, 2002.

[Bol04] Bela Bollobas. *Extremal Graph Theory*. Dover Publications, Incorporated, 2004.

[Cha00] Bernard Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *J. ACM*, 47(6):1028–1047, 2000.

[CLR90] T. H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 1990.

[CV86] Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Inf. Control*, 70(1):32–53, 1986.

[Elk04a] M. Elkin. A faster distributed algorithm for constructing a mnimum spanning tree. In *Proc. ACM-SIAM Symp. on Discr. Algorithms*, pages 352–361, 2004.

[Elk04b] M. Elkin. Unconditional lower bounds on the time-approximation tradeoff of the distributed minimum spanning tree problem. In *Proc. 36th Annual ACM Symp. on Theory of Computing*, pages 331–340, 2004.

[ES46] P. Erdős and A. H. Stone. On the structure of linear graphs. *Bull. Am. Math. Soc.*, 52:1089–1091, 1946.

[GGST86] H. Gabow, Z. Galil, T. Spencer, and R. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(1):109–122, 1986.

[GPS87] A. Goldberg, S. Plotkin, and G. Shannon. Parallel symmetry-breaking in sparse graphs. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 315–324, New York, NY, USA, 1987. ACM Press.

[KMNW05] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Fast deterministic distributed maximal independent set computation on growth-bounded graphs. In P. Fraigniaud, editor, *Distributed Computing: 19th International Conference, DISC 2005, Cracow, Poland*, volume 3724 of *Lecture Notes in Computer Science*, pages 273–283, Berlin, November 2005. Springer Verlag.

[KMW04] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing newly deployed ad hoc and sensor networks. In *MOBICOM '04: Proceedings of the tenth International Conference on Mobile Computing and Networking*, pages 7–14, 2004.

[KMW05] F. Kuhn, T. Moscibroda, and R. Wattenhofer. On the locality of bounded growth. In *PODC '05: Proceedings of the twenty-fourth annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 60–68, New York, NY, USA, 2005. ACM Press.

[KP98] Shay Kutten and David Peleg. Fast distributed construction of small k-dominating sets and applications. *J. Algorithms*, 28(1):40–66, 1998.

[Lin92] Nathan Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.

[LPS88] A. Lubotzky, R. Phillips, and P Sarnak. Ramanujan graphs. *Combinatorica*, 8:261–277, 1988.

[Mar88] G. A. Margulis. Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators. (russian). *Problemy Peredachi Informatsii*, 24:51–60, 1988.

[MvRW06] T. Moscibroda, P. von Rickenbach, and R. Wattenhofer. Analyzing the energy-latency trade-off during the deployment of sensor networks. In *INFOCOM '06: Proceedings of the 25th IEEE Conference on Computer Communications*, pages 7–14, 2006.

[PR00] Seth Pettie and Vijaya Ramachandran. An optimal minimum spanning tree algorithm. In *Automata, Languages and Programming*, pages 49–60, 2000.

[Soa92] J. Soares. Graph spanners: a survey. *Congress Numerantium*, 89:225–238, 1992.

[WL99] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *DIALM '99: Proceedings of the third International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 7–14, 1999.