Fast Distributed Network Decompositions and Covers [*]

Baruch Awerbuch [†] Bonnie Berger [‡] Lenore Cowen [§]
David Peleg [¶]

September 14, 1995

Abstract

This paper presents deterministic sublinear-time distributed algorithms for *network decomposition* and for constructing a *sparse neighborhood cover* of a network. The latter construction leads to improved distributed preprocessing time for a number of distributed algorithms, including all-pairs shortest paths computation, load balancing, broadcast, and bandwidth management.

[*]A preliminary version of this paper appeared in the Proceedings of the *Eleventh Annual ACM Symposium on the Principles of Distributed Computing.*

[†]Lab. for Computer Science, MIT, Cambridge, MA 02139. Supported by Air Force Contract AFOSR F49620-92-J-0125, NSF contract 9114440-CCR, DARPA contracts N00014-91-J-1698 and N00014-J-92-1799, and a special grant from IBM.

[‡]Dept. of Mathematics and Lab. for Computer Science, MIT. Supported in part by an NSF Postdoctoral Research Fellowship and an ONR grant provided to the Radcliffe Bunting Institute.

[§]Dept. of Math Sciences, Johns Hopkins University, Baltimore, MD 21218. This work was done while the author was a grad student in the Dept. of Mathematics at MIT and Supported in part by Air Force Contract OSR-89-02171, Army Contract DAAL-03-86-K-0171 and DARPA Contract N00014-91-J-1698

[¶]Department of Applied Mathematics and Computer Science, The Weizmann Institute, Rehovot 76100, Israel. Supported in part by an Allon Fellowship, by a Bantrell Fellowship, by a Walter and Elise Haas Career Development Award and by the Israel Basic Research Foundation.

1 Introduction

1.1 Background

This paper is concerned with fast deterministic algorithms for constructing network decompositions in the distributed network model, particularly a strong form of network decomposition known as a *sparse neighborhood cover.* Given an undirected weighted graph (or network), the *t-neighborhood* of a vertex v, for $t \geq 0$, is the collection of nodes within distance t away from v in the graph. A *t-neighborhood cover* is a set of overlapping "clusters" of nodes in the graph with the property that for any node, there exists a cluster in the cover that contains its t-neighborhood. A t-neighborhood cover is called *high-quality* or *sparse* (defined formally in Section 2) when it achieves an optimal tradeoff between the maximum diameter and the maximum cluster overlap.

Such a cover efficiently represents the local neighborhoods in a graph. In fact, the method of representing networks by *sparse neighborhood covers*, introduced in [Pel89], has recently been identified as a key to the modular design of efficient network algorithms [Pel93, AP90b, AP90a, AR91, BFR92]. Using this method as a basic building block leads to significant performance improvements for several fundamental network control problems (such as shortest paths [AR91], job scheduling and load balancing [AKP92], broadcast and multicast [ABP91], routing with small tables [AP92], deadlock prevention [AKP91], bandwidth management in high-speed networks [ACG$^+$90], and database management [BFR92]), as well as for classical problems in sequential computing (such as finding small edge cuts in planar graphs [Rao92] and approximate all-pairs shortest paths [ABCP93, Coh93]). In most of these applications, sparse neighborhood covers yield a polylogarithmic-overhead solution to the problem. Thus, in a sense, the impact of efficient sparse neighborhood cover algorithms on distributed network algorithms is analogous to the impact of efficient data structures (like balanced search trees or 2-3 trees) on sequential computation.

In parallel to the introduction of sparse neighborhood covers in [Pel89], the strongly related, yet distinct, notion of network decomposition was introduced in [AGLP89]. The main difference between the two notions is that sparse neighborhood covers, as used in [Pel93, AP90b, AP90a, BFR92, AKP92, AKP91], consist of clusters that capture the expected definition of local neighborhood (see Section 2), while network decomposition as utilized in [AGLP89, LS91, PS92, ABCP92] employs only a weak notion of neighborhood. In fact, the way network decomposition is defined in [AGLP89], the clusters might not even be internally connected. For many applications in distributed computing, the stronger notion of sparse covers is essential. For example, decompositions based on the network decomposition structure introduced in [AGLP89] are not sufficient to support local routing, where the path between two vertices in the same cluster should consist entirely of vertices within that cluster. Notable exceptions for which such a decomposition suffices are those applications based on "symmetry-breaking"; i.e., such a decomposition can be used to construct a maximal independent set or a $(\Delta+1)$ coloring (where Δ is the maximum vertex degree in the graph) fast in the distributed domain [AGLP89, LS91, PS92].

The goal of this paper is fast distributed algorithms for both high-quality network decomposition, and sparse neighborhood covers.

1.2 Previous work

Previous work on the construction of neighborhood covers and decompositions can be classified into three main groups, to be discussed next. First, a fast algorithm is given in [AGLP89] for obtaining a network decomposition with $O(n^\epsilon)$-diameter clusters, for $\epsilon = O(\sqrt{\log\log n}/\sqrt{\log n})$. This algorithm requires $O(n^\epsilon)$ time in the distributed setting, and $O(nE)$ sequential operations.

Unfortunately, not only are the constructions of [AGLP89] restricted to network decompositions (rather than the stronger construct of neighborhood covers), but they are also inefficient in terms of the *quality* of the decomposition (measured, as discussed earlier, by the maximum diameter-overlap tradeoff). Roughly speaking, the inefficiency factor is $O(n^\epsilon)$, and this factor carries over to all but some of the graph-theoretic applications, rendering the decompositions of [AGLP89] expensive in a number of practical contexts. These constructions are, nonetheless, sufficient for the two main applications mentioned above, namely, the maximal independent set problem and $(\Delta+1)$ coloring. We have already remarked that a network decomposition suffices for these applications. There is also no penalty for the poor quality of the clusters because to construct an MIS or a $(\Delta+1)$ coloring, one needs to traverse the $O(n^\epsilon)$-diameter clusters only a constant number of times. In contrast, network control applications, such as routing, online tracking of mobile users, and all-pairs shortest paths, require us to traverse the clusters many times. Therefore, a higher-quality decomposition is needed to avoid a large blowup in the running time for these latter applications.

A second group of constructions consists of (sequential) greedy algorithms for the construction of sparse neighborhood covers [Pel89, AP90b, LS91]. These algorithms yield the desired optimal tradeoff, and their bounds apply to all alternate notions of network decomposition or sparse neighborhood covers. (In fact, the algorithm of [AP90b] is more general. It can create a subsuming cover for an *arbitrary* initial collection of clusters, and not necessarily to the collection of 1-neighborhoods. In addition, it applies also to the weighted case, with the corresponding weighted notions of distance and diameter.) However, these algorithms are inherently sequential, and their distributed implementation requires $O(n \log n)$ time.

The third group consists of randomized distributed algorithms. The algorithm of [LS91] achieves a high-quality network decomposition with high probability by introducing randomization, and is very efficient. However, the algorithm does not yield a sparse neighborhood cover. Furthermore, even when a network decomposition suffices instead of a neighborhood cover, since we are concerned here with using the clusters as a data structure and running various applications on top of it, a randomized solution might not be acceptable in some cases. This is especially true because one cannot just run a randomized distributed algorithm several times to guarantee a good underlying decomposition, since it is impossible to efficiently check the global quality of the decomposition in the distributed model. Thus a fast deterministic algorithm that *guarantees* a good underlying set of clusters is of more than theoretical interest.

In summary, all three previous approaches fell short of achieving the goal of constructing a *high-quality* network decomposition (or a sparse neighborhood cover) *deterministically* and in *sublinear time;* each achieves two of the desired properties at the expense of the third.

1.3 Contents of this paper

In this paper we achieve the goal of constructing a high-quality network decomposition (or a sparse neighborhood cover), deterministically in sublinear time. We construct all alternate notions of network decomposition, or sparse neighborhood covers in time $O(n^\epsilon)$, for $\epsilon = O(1/\sqrt{\log n})$. The reduction from time $O(n^\epsilon)$ for $\epsilon = O(\sqrt{\log \log n}/\sqrt{\log n})$ to $\epsilon = O(1/\sqrt{\log n})$ is achieved directly from the corresponding speedup for the [AGLP89] construction due to [PS92]. In addition we present a randomized algorithm that constructs all notions of high-quality network decomposition in polylogarithmic expected time, including the useful sparse neighborhood covers.

Our results are presented in terms of Linial's model for static synchronous networks (see Section 3.4), but can be adapted to a more realistic dynamic asynchronous environment using the existing transformer techniques of [AAG87, AP90b, APPS92].

2 Notions of network decomposition

In this section we survey the different formulations of network decomposition, and discuss their relations. Within each family of definitions, we also discuss what it means to have a *high-quality* decomposition or cover, in terms of the optimal tradeoffs between low diameter and sparsity.

Our definitions consider a graph $G = (V, E)$ whose vertices are grouped into a collection of (possibly overlapping) sets $S_1, \ldots, S_r$ (called also *clusters*). These clusters need not necessarily be internally connected in the graph. This collection is referred to as a *cover*. For the special case that the clusters are disjoint, we refer to the cluster collection as a *decomposition*.

Our notion of *distance* in a graph is the usual (unweighted) one, i.e., the distance between $u, v \in V$ in G, denoted $dist_G(u, v)$, is the length (in edges) of the shortest path between u and v in G. The distance between two clusters S, S' is analogously defined to be the minimum distance (in G) between any two vertices $v \in S$ and $v' \in S'$. A collection of clusters is said to be λ-*separated* if every two clusters in it are at least distance $\lambda + 1$ apart.

However, we are also interested in distances *inside* clusters, and for that purpose, we must distinguish between two distinct notions.

Definition 2.1 The *weak* distance between $u, v \in S_i$ is simply their distance in G, $dist_G(u, v)$. Namely, the path connecting them is allowed to shortcut through vertices not in S_i. The *weak diameter* of S_i is defined as

$$diam(S_i) = \max_{u,v \in S_i} (dist_G(u, v)).$$

Similarly, the *weak radius* of S_i is defined as

$$rad(S_i) = \min_{u \in S_i} \max_{v \in S_i} (dist_G(u, v)).$$

The *strong* distance between $u, v \in S_i$, denoted $dist_{S_i}(u, v)$, is the length of the shortest path between u and v, on the subgraph of G induced by S_i. Namely, all vertices on the path connecting u and v must also be in S_i. The *strong diameter* of S_i is defined as

$$Diam(S_i) = \max_{u,v \in S_i} (dist_{S_i}(u, v)).$$

The *square* of the graph G, denoted G^2, is defined to be the two-step transitive closure of G. I.e., G^2 contains an edge (u, v) if either this edge is in G itself, or there exists an intermediate vertex w s.t. (u, w) and (w, v) are in G. Similarly, G^t is the transitive closure of G to distance t, i.e., it contains an edge between any two vertices that are connected by a path of length t or less in G.

The *t-neighborhood* of a vertex $v \in V$ is defined as $N_t(v) = \{w \mid dist_G(w, v) \leq t\}$. This notion is extended to sets of vertices, defining the t-neighborhood of a set W to be $N_t(W) = \bigcup_{v \in W} N_t(v)$. (Note that always $W \subseteq N_t(W)$.)

We are now ready to define the alternate notions of network decompositions and covers. First, we give the weak diameter definition, which is equivalent to the definitions in [AGLP89, LS91, PS92].

Definition 2.2 [Weak Network Decomposition.] For an undirected graph $G = (V, E)$, a *weak* (χ, d, λ)*-decomposition* is defined to be a χ-coloring of the vertices of the graph, i.e., a mapping $\psi : V \mapsto \{1, \ldots, \chi\}$, that satisfies the following properties:

1. each color class $\psi^{-1}(i)$ is partitioned into (an arbitrary number of) disjoint vertex clusters $C_1^i, \ldots, C_{l_i}^i$;
2. the *weak* diameter of any cluster C_j^i of a single color class $\psi^{-1}(i)$ satisfies $diam(C_j^i) \leq d$;
3. each collection of monochromatic clusters is λ-separated.

The *quality* of the decomposition is measured by the tradeoffs between the parameters d, λ and χ. It is known that there are graphs for which χ must be $\Omega(kn^{1/k})$ to achieve a weak decomposition into clusters of diameter bounded by $O(k\lambda)$ and separation λ [LS91]. Consequently, a weak (χ, d, λ)-decomposition is said to be *high-quality* if it achieves the optimal tradeoff; namely, when $d = O(k\lambda)$, and the coloring number χ is at most $kn^{1/k}$, for some $k \geq 1$.

Typically, we are most concerned with the case of a high-quality weak decomposition when χ and d are both $O(\log n)$ (which occurs, for constant λ, when $k = \Theta(\log n)$).

We may occasionally refer to a (monochromatic) cluster C whose vertices are colored by some color j, as being colored j itself, and write $\psi(C) = j$.

A simple variant of weak network decomposition yields the related notion of a *strong* network decomposition.

Definition 2.3 [Strong Network Decomposition.] For an undirected graph $G = (V, E)$, a *strong* (χ, d, λ)*-decomposition* is defined just as a weak (χ, d, λ)-decomposition, except that in Property 2, we substitute "strong" for "weak" diameter.

As with the weak-diameter definition, the "high-quality" tradeoffs are optimal. A strong (χ, d, λ)-decomposition can be thought of as a generalization of the standard graph coloring problem, where χ is the number of colors used, and the clusters are super-vertices of strong-diameter d.

Clearly, any strong network decomposition is also a weak decomposition, but the converse is not necessarily true. All the results in [AGLP89, LS91, PS92, ABCP92] are stated in terms of weak network decomposition, but some can be extended to strong network decomposition (See [Cow93] for a survey.).

We now present the definition for sparse neighborhood covers. Notice that this is a strong diameter definition.

Definition 2.4 A (k, t, Δ)-*neighborhood cover* is a collection of clusters $S_1, \ldots, S_r$, with the following properties:

1. For every vertex v, there exists a cluster S_i s.t. $N_t(v) \subseteq S_i$.
2. The strong diameter of each cluster S_i satisfies $Diam(S_i) \leq O(kt)$.
3. Each vertex belongs to at most Δ clusters.

Analogously to the case of decompositions, the quality of the cover is measured in terms of the tradeoffs between the parameters Δ, t and k. It is known that there are graphs for which Δ must be $\Omega(kn^{1/k})$ to achieve a cover of all radius t neighborhoods $N_t(v)$ by clusters of strong diameter bounded by $O(kt)$. Consequently, a (k, t, Δ)-neighborhood cover is said to be *sparse,* if $\Delta \leq kn^{1/k}$.

The parameter k is bounded between two natural extreme cases. Setting $k = 1$, the set of all balls $N_t(v)$ of radius t around each vertex v is a sparse neighborhood cover. In this case, the diameter of a ball is t, but each vertex might appear in every ball, so Δ may be as high as n. On the other extreme, setting $k = Diam(G)/t$, the single cluster composed of the entire graph G is a sparse neighborhood cover. In this case, each vertex appears only in a single cluster G, so the degree bound is $\Delta = 1$, but the strong diameter of the cover equals the diameter of G, which could be as high as n.

A natural breakpoint is obtained by setting $k = \log n$ (the typical and useful setting, for most of the applications we are interested in). This yields a sparse $(\log n, t, O(\log n))$-neighborhood cover, which is a collection of clusters S_i with the following properties:

- the clusters contain all t-neighborhoods $N_t(v)$,

- the diameter of each cluster is bounded by $O(t \log n)$, and
- each vertex is contained in at most $c \log n$ clusters, for constant $c > 0$.

We remark that this bound is tight to within a constant factor; there exist graphs for which any $(\log n, t, \Delta)$-neighborhood cover has $\Delta = \Omega(\log n)$, i.e., it places some vertex in at least $\Omega(\log n)$ sets [LS91]. When $k = \log n$, we find that sparse neighborhood covers form a useful data structure to *locally* represent the t-neighborhoods of a graph.

The new deterministic distributed algorithm described in Section 4 constructs deterministically a sparse neighborhood cover, and can easily be transformed into an algorithm for constructing a (strong, and therefore also weak) diameter decomposition.

3 Weak network decomposition

3.1 Outline

In this section, we introduce the new distributed algorithm `Color`, which recursively builds up a weak $(kn^{1/k}, 2k+1, 1)$-decomposition, for a parameter k. It invokes the procedure `Compress`, which, in turn, runs the procedure `Greedy_Color`, which is a modified version of the greedy algorithm of [AP92] on separate clusters. Note that all distances in the discussion below, including those in the same cluster, are assumed to be *weak* distances (i.e., distances in the graph G).

`Color` is implicitly taking higher and higher powers of the graph. The straightforward but crucial observation on which the power graph approach is based is the following:

Lemma 3.1 A $(\chi, d, 1)$-decomposition on G^t is a (χ, dt, t)-decomposition on G. $\Box$

Choosing t well at the top level of the recursion, guarantees that vertices in different clusters of the same color are always separated by at least twice their maximum possible radii. We can thus use procedure `Greedy_Color` to *recolor* these separate clusters in parallel without collisions.

The recursive algorithm has two parts:

1. Find a weak (χ, dt, t)-decomposition, where $\chi = xkn^{1/k}$ and $d, t = 2k+1$, on each of x disjoint subgraphs. Here x is a parameter of the form 2^y, to be determined later on.
2. Merge these together by recoloring, to get a weak $(kn^{1/k}, 2k+1, 1)$-decomposition.

Note that in essence, the decomposition generated in the first step is utilized in the "traditional" way, for symmetry-breaking, in order to facilitate the construction of the improved decomposition in step 2. The fact that the clusters generated in the first step are separated to distance $2k+1$, coupled with the fact that Step 2 invokes a sequential algorithm in parallel inside each cluster separately, and this algorithm constructs clusters of radius at most k, ensures that these parallel invocations do not interfere.

3.2 The coloring algorithm

In this subsection we present the coloring algorithm `Color`, its main procedure `Compress` and its sub-procedure `Greedy_Color`.

3.2.1 Procedure `Greedy_Color`

Let us first describe procedure `Greedy_Color`. Procedure `Greedy_Color` is a weak-diameter variant of the (sequential) procedure used in the greedy algorithm of [AP92] for determining, in each iteration, what vertices will be colored in that iteration. The procedure receives a vertex set R. It then constructs a set DR of vertices in the graph with the following properties. First, $DR \subseteq R$. Secondly, DR is the union of a collection of 1-separated clusters $\{C_1, \ldots, C_l\}$, of radius at most k, whose centers are nodes of R. Thirdly, DR contains at least a $1/|R|^{1/k}$ fraction of the vertices of $N_1(DR) \cap R$. These clusters will be colored in a single color by procedure `Compress`.

The procedure operates as follows. The procedure picks an arbitrary vertex v in R (called the *center* vertex), and grows a ball of vertices DR around it of the smallest radius r such that the vertices from the set R in DR are at least a $1/|R|^{1/k}$ fraction of the vertices from the set R in the ball of radius $r+1$ around v, $N_1(DR)$. It is easy to prove that there always exists an $r \leq k$ for which this condition holds. Then DR is made into a cluster, and the nodes of $N_1(DR) \cap R$ are eliminated from the graph. Then another arbitrary vertex of R is picked, and the process is repeated, until no more vertices are left in R.

Procedure `Greedy_Color`(R)

Input: A cluster R.
Output: A set $DR \subseteq R$ containing at least $|R|^{1-1/k}$ vertices of R.

```
1. $DR \leftarrow \emptyset$; $\hat{R} \leftarrow R$.
```

```
2. While $\hat{R} \neq \emptyset$ do:
    (a) $S \leftarrow \{v\}$ for some $v \in \hat{R}$.
    (b) While $|N_1(S) \cap \hat{R}| > |R|^{1/k}|S|$ do:
            $S \leftarrow S \cup (N_1(S) \cap \hat{R})$.
    (c) $DR \leftarrow DR \cup S$.
    (d) $\hat{R} \leftarrow \hat{R} - (N_1(S) \cap \hat{R})$.
```

3.2.2 Procedure `Compress`

Let us next describe procedure `Compress`, whose role is to take an initial legal coloring ψ_{old} that constitutes a weak decomposition with "many" (specifically, $xkn^{1/k}$) colors and "large" separation (specifically, $2k+1$), and compress it into a new coloring ψ_{new} providing a weak decomposition with fewer (specifically, $kn^{1/k}$) colors and separation 1, using sub-procedure `Greedy_Color`.

The procedure `Compress` operates in $kn^{1/k}$ iterations, each of which colors a fraction of the old-colored vertices remaining with a new color i. Each iteration i looks separately at each cluster C of each old color-class j. For each such cluster, it constructs the set R of nodes in $N_k(C)$ that have not yet been colored by a new color, and then activates procedure `Greedy_Color`, which returns a set DR consisting of at least $|R|^{1/k}$ vertices of R, arranged in clusters of weak-radius at most k. The crucial property of these clusters is that they are 1-separated, both from each other and from the rest of the nodes yet to be colored, since they are chosen as the *interiors* of some balls constructed by the procedure. This is why Procedure `Compress` can color all the nodes of all the sets DR constructed for each cluster C of each old color-class j by the same new color i. (Procedure `Compress` actually processes the clusters of each old color j sequentially, but 1-separation is still ensured, since after coloring the interior of a certain ball by new color j, the procedure removes all the uncolored vertices of the *entire* ball from the set of nodes to be colored W.)

Procedure `Compress`(G, ψ_{old})

Input: A graph G, and a weak $(xkn^{1/k}, (2k+1)^2, 2k+1)$-decomposition ψ_{old} on G.
Output: A weak $(kn^{1/k}, 2k+1, 1)$-decomposition ψ_{new} on G.

```
$P \leftarrow V$.          /* The set of vertices yet without a new color. */
For $i = 1$ to $kn^{1/k}$ do (sequentially):          /* Generate new color $i$. */
```

```
1. $W \leftarrow P$.
2. For $j = 1$ to $xkn^{1/k}$ do (sequentially):        /* Cycle through old-color classes. */
   In parallel for each cluster $C$ colored $\psi_{old}(C) = j$ do:
     (a) Elect a leader for cluster $C$.
     (b) The leader learns the set $R = N_k(C) \cap W$.
     (c) The leader executes locally $DR \leftarrow$ Greedy_Color($R$).
     (d) Color the vertices $v \in DR$ with new color $\psi_{new}(v) = i$.
     (e) Set $P \leftarrow P - DR$.
     (f) Set $W \leftarrow W - R$.
```

The fact that procedure `Compress` colors at least $|R|^{1/k}$ vertices of R in each neighborhood it processes, is later used in the analysis in order to prove that after $kn^{1/k}$ iterations, the set P becomes empty.

Let us comment that it might possibly happen that the set P becomes empty earlier than that, in which case all subsequent iterations will do nothing. In the sequential algorithms for decomposition or cover construction, such as that of [AP90b] for instance, this problem is bypassed by using a conditional loop, repeated only until all vertices are colored. Unfortunately, this condition is not easy to detect in a distributed fashion, so a distributed algorithm must do without it.

3.2.3 Algorithm `Color`

We finally present the entire recursive algorithm `Color`.

Algorithm `Color`(G)

Input: A graph $G = (V, E)$, $|V| = n$, and integer $k \geq 1$.
Output: A $(\chi, 2k+1, 1)$-decomposition of G, $\psi : V \mapsto \{1, \ldots, \chi\}$, for $\chi = kn^{1/k}$.

```
1. Compute $G^{2k+1}$.
2. If $G$ has less than $x$ vertices, then run the simple greedy algorithm of [AP92, LS91] to
   generate a $(kn^{1/k}, 2k+1, 1)$-decomposition for $G$, and return.
3. Partition the vertices of $G$ into $x$ subsets, $V_1, \ldots, V_x$ (based on the last $\log x$ bits of
   vertex IDs, which are then discarded).
```

```
4. Define $G_i$ to be the subgraph of $G^{2k+1}$ induced on $V_i$.

5. **In parallel** for $i = 1, \ldots, x$ **do:**
   $\psi_i \leftarrow$Color($G_i$).                /* recursive application */

6. **For** each $v \in V$ **do:**
   If $v \in V_i$ then color $v$ with the color $\psi_R(v) \leftarrow \langle i, \psi_i(v) \rangle$.

7. $\psi \leftarrow$ Compress($\psi_R$)
```

3.3 Correctness and Analysis

Let us first establish the basic properties of Procedure `Greedy_Color`.

Lemma 3.2 The set DR constructed by Procedure `Greedy_Color` satisfies the following properties:

(1) $DR \subseteq R$,

(2) DR is the union of a 1-separated cluster collection $C_1, \ldots, C_l$ with weak-radius $rad(C_i) \leq k$,

(3) $N_1(DR) \cap R = R$, and

(4) $|DR| \geq |R|^{1-1/k}$.

Proof Claim (1) is immediate from the procedure.

Let $C_1, \ldots, C_l$ be the clusters generated by the procedure and added to the set DR. That this collection is 1-separated follows from the fact that whenever a cluster C_i is added to the set DR, the R vertices in its 1-neighborhood are removed from $\hat{R}$, hence won't participate in any clusters constructed later.

We prove the weak-radius bound on the clusters by contradiction. Suppose that some cluster $S = C_i$ has radius strictly greater than k. Then through k successive iterations of **While** loop (2b) of the procedure, $|N_1(S) \cap \hat{R}| > |R|^{1/k}|S|$. In each iteration, S is increased to $N_1(S) \cap \hat{R}$ and thus grows by an $|R|^{1/k}$ factor. After k such iterations, $|S| > |R|$; contradiction, since $S \subseteq R$ by construction.

Claim (3) follows from the fact that at the end of the procedure's execution, R becomes empty, and every vertex erased from it belongs to $N_1(DR)$.

Finally, we prove Claim (4). For each new cluster C_i in DR, let $Q_i = N_1(C_i) \cap \hat{R}$. By the stopping condition on the **While** loop (2b) in procedure `Greedy_Color`, we know that

$|Q_i| \leq |R|^{1/k}|C_i|$. Note that the sets C_i in DR are disjoint, and moreover, the sets Q_i are disjoint as well, since each set Q_i is immediately removed from $\hat{R}$. Since $DR = \bigcup_i(C_i)$ and $R = \bigcup_i(Q_i)$, summing over all the sets C_i, we get that $|DR| \geq |R|/|R|^{1/k}$. Claim (4) follows. □

Let us now turn to analyzing the properties of Procedure `Compress`. Let $P(i)$ denote the set of vertices that remain uncolored at the end of the ith (main) iteration of procedure `Compress`. We claim that this set shrinks by the desired factor, or formally:

Lemma 3.3 $|P_i| \leq |P_{i-1}| - |P_{i-1}|^{1-1/k}$.

Proof Consider iteration i. Note that at its end, the set W becomes empty. This is because every uncolored node v belongs to some old color class j, and assuming v was not colored in any internal iteration $k < j$, in the jth internal iteration v will join R, and consequently be removed from W.

Consider the operation of the procedure on old color j and old cluster C. Let $R(j,C)$ denote the set $R = N_k(C) \cap W$ computed for this pair, and let $DR(j,C)$ denote the set removed from P in step (2e) for this pair. By Lemma 3.2,

$$|DR(j,C)| \geq |R(j,C)|^{1-1/k}. \qquad (1)$$

We claim that every two sets $R(j,C)$ and $R(j',C')$ are disjoint. When $j' = j$ the claim follows from the fact that the clusters of a given old color j are $2k+1$-separated, hence the corresponding sets $N_k(C)$ and $N_k(C')$ are disjoint. When $j' > j$ the claim follows from the fact that $R(j,C)$ is subsequently removed from W. Similarly, every two sets $DR(j,C)$ and $DR(j',C')$ are disjoint.

Since the set W becomes empty in the end, $P_{i-1} = W = \bigcup_{j,C} R(j,C)$. Denote the set of nodes that were colored in iteration i by $Z = \bigcup_{j,C} DR(j,C)$. By the above disjointness arguments we have $|W| = \sum_{j,C}|R(j,C)|$ and $|Z| = \sum_{j,C}|DR(j,C)|$. Since $a^\alpha + b^\alpha \geq (a+b)^\alpha$ for $\alpha < 1$, we get by (1) that satisfies $|Z| \geq |W|^{1-1/k}$, hence $P_i \leq |W| - |W|^{1-1/k}$. □

Corollary 3.4 After $kn^{1/k}$ (main) iterations of Procedure `Compress`, all the sets P_j, hence also the set P, become empty. □

Lemma 3.5 Given a graph G and a weak $(xkn^{1/k}, (2k+1)^2, 2k+1)$-decomposition ψ_{old} on the vertices of G as input, the output of Procedure `Compress` is a weak $(kn^{1/k}, 2k+1, 1)$-decomposition ψ_{new} on G.

Proof Consider some iteration i of Procedure `Compress`. For each old color j and old cluster C, let $DR(j,C)$ denote the set removed from P in step (2e) for this pair. We first need to argue that the collection of clusters $DR(j,C)$ that were generated (and colored i) in iteration i is 1-separated.

Look at two sets $DR(j,C)$ and $DR(j',C')$. When $j'=j$ the claim is clear, since $DR(j,C) \subseteq N_k(C)$ and $DR(j',C') \subseteq N_k(C')$, and the clusters of the old color class j were $2k+1$-separated. When $j<j'$, 1-separation is guaranteed since once $DR(j,C)$ was colored i, all the uncolored vertices in its 1-neighborhood, $N_1(DR(j,C)) \cap \hat{R}$, were removed from the set of candidate nodes to be colored, W.

The bound of $2k+1$ on the diameter of each new cluster is immediate from Lemma 3.2.

Finally, the bound on the number of new colors follows from the previous lemma. □

Lemma 3.6 The running time of procedure `Compress` is $x(kn^{1/k})^2(2k+1)^2$.

Proof Overall, there are $xkn^{1/k} \cdot kn^{1/k}$ internal iterations, and the number of steps per internal iteration is proportional to the diameter of the k-neighborhoods $N_k(C)$ (which is the area scanned by C's leader). Since clusters in the initial decomposition are of diameter $(2k+1)^2$, the relevant area has diameter $(2k+1)^2$, hence the total time is $x(kn^{1/k})^2(2k+1)^2$. □

Finally, we analyze the properties of Algorithm `Color` itself.

Lemma 3.7 The coloring ψ_R produced in Step 6 of the algorithm (based on the recursion output) is a weak $(xkn^{1/k},(2k+1)^2,2k+1)$-decomposition of G, and the coloring ψ produced in Step 7 of the algorithm is a weak $(kn^{1/k},2k+1,1)$-decomposition of G.

Proof By induction on the level of recursion. The lowest level is taken care of by Step 2 of the algorithm. Assuming, by inductive hypothesis, that each coloring ψ_i returned in Step 5 of the algorithm is a weak $(kn^{1/k},2k+1,1)$-decomposition of G, it follows that their combination ψ_R is a weak $(xkn^{1/k},2k+1,1)$-decomposition of G^{2k+1}, and by Lemma 3.1 this is also a weak $(xkn^{1/k},(2k+1)^2,2k+1)$-decomposition of G. Finally, by Lemma 3.5, Step 7 of the algorithm yields a weak $(kn^{1/k},2k+1,1)$-decomposition of G. □

It remains to analyze the running time of the algorithm `Color`. We observe the following. First, the branching phase of the recursion takes time $T'(n) \leq (2k+1)T'(n/x)+x$. By Lemma 3.6, the merge takes time $x(kn^{1/k})^2(2k+1)^2$. Overall, we have

$$\begin{aligned} T(n) &\leq (2k+1)T(n/x)+x(kn^{1/k})^2(2k+1)^2 \\ &\leq (2k+1)^{\log n/\log x} x(kn^{1/k})^2(2k+1)^2. \end{aligned}$$

This bound is optimized by selecting $x = 2^{\sqrt{\log n}\sqrt{1+\log k}}$. We get the following.

Lemma 3.8 Fixing $x = 2^{\sqrt{\log n}\sqrt{1+\log k}}$, the running time of algorithm `Color` is $n^{2\sqrt{1+\log k}/\sqrt{\log n}+2/k}(2k+1)^2$.

Theorem 3.9 There is a deterministic distributed algorithm which given a graph $G = (V, E)$, finds a weak $(kn^{1/k}, 2k+1, 1)$-decomposition of G in $n^{2\sqrt{1+\log k}/\sqrt{\log n}+2/k}(2k+1)^2$ time.

Corollary 3.10 There is a deterministic distributed algorithm which given $G = (V, E)$, finds a weak $(O(\log n), O(\log n), 1)$-decomposition of G in $n^{3\sqrt{\log\log n}/\sqrt{\log n}}$ time, which is in $O(n^\epsilon)$ for any $\epsilon > 0$.

Independently, and at the same time as we introduced the above algorithm, Panconesi and Srinivasan [PS92] obtained a slightly better asymptotic running time for a low-quality weak network decomposition than that achieved by Awerbuch et. al. [AGLP89]. As they remark, using a version of our *transformer* algorithm (see the next section), gives the same improvement in running time for the construction of a *high-quality* weak network decomposition. We thus obtain the following corollary: .

Corollary 3.11 There is a deterministic distributed algorithm which given $G = (V, E)$, finds a weak $(O(\log n), O(\log n), 1)$-decomposition of G in $O(n^{O(1/\sqrt{\log n})})$ time, which is in $O(n^\epsilon)$ for any $\epsilon > 0$.

3.4 Distributed implementation

Finally, let us discuss the distributed implementation of the algorithm. The distributed model of computing we will be concerned with, hereafter referred to as the *free* model, is due to Linial [Lin87]. Much as PRAM algorithms in parallel computing gives a good indication of parallelism, the free model gives a good indication of locality and distributed time.

In the free distributed model, the underlying network topology is represented by a graph, $G = (V, E)$, where there is a processor at each vertex, and there is an edge between two vertices if and only if there is a direct communication link between the corresponding processors. Communication is completely synchronous, and reliable. Every time unit, each processor may pass messages to each of his neighbors. There is no limit on the size of these messages. Also, we do not charge for the time that it requires individual processors to compute functions; we only require that these are polynomial time computations. Hence all operations within a subgraph of weak-diameter t can be performed centrally by collecting information to the leader and doing them locally (at no cost), and hence require time t.

Some implementation issues still need to be discussed. Procedure `Color` in effect works on higher and higher powers of the graph. Notice that to implement the (logical) graph G^t

in the (actual) distributed network G, we might have to traverse paths of length t in order to look at all our neighbors in the graph G^t, since the only edges physically available in the network are those of the underlying graph G. Therefore the time for running an algorithm on the graph G^t blows up by a factor of t.

One last technical point to be discussed concerns the way clusters are handled as single nodes (for coloring etc.) in the recursive algorithm. This is done by electing a "leader" vertex for each cluster, which does all the computation for its cluster.

We remark that direct practical implementation of distributed algorithms on real distributed systems may limit the size of messages that can be sent in one step over an edge, charge for local space and time, remove the assumption that the system is synchronous, and might also seek to handle faulty processors, or a dynamically changing network. Because we are presenting the first sub-diametric time algorithms for high-quality network decompositions and covers, we have chosen in this paper to work in the mathematically cleanest model. For ideas on how one might go about adapting the algorithms in this paper to deal with some of the above concerns, the reader is referred to the *transformer* techniques in [AAG87, AP90b, APPS92].

4 Sparse neighborhood covers

4.1 Outline

We now turn to algorithms for generating sparse neighborhood covers. We introduce an algorithm `Sparse`, which takes as input parameters k and t, and outputs a $(k, t, kn^{1/k})$-neighborhood cover for a given network G.

Algorithm `Sparse` invokes two procedures, named `Cover` and `Decomp`. Procedure `Cover` is a modification of the sequential sparse cover procedure of Awerbuch and Peleg [AP90b]. Procedure `Decomp` can be any existing procedure which given a graph $G = (V, E)$, finds a weak $(\chi_{old}, d_{old}, 1)$-decomposition of G.

Lemma 4.1 [AP90b] There exists a procedure `Cover`(R) that given a graph $G = (V, E)$, $|V| = n$, a collection of vertices R and an integer k, constructs a set of vertices DR and a collection of clusters $\mathcal{DU}$, satisfying the following properties:

(1) For every $v \in DR$, the t-neighborhood $N_t(v)$ is contained in some cluster in $\mathcal{DU}$.

(2) $Y \cap Y' = \emptyset$ for every $Y, Y' \in \mathcal{DU}$,

(3) $|DR| \geq |R|^{1-1/k}$, and

(4) The (strong) diameter of $\mathcal{DU}$ clusters satisfies $\max_{T\in\mathcal{DU}} Diam(T) \leq (2k-1)\cdot 2t$.

(Let us remark that it is possible to introduce slight modifications to Procedure `Cover`(R) of [AP90b], that will result in improving the constant factor occuring in the exponent of the expression for the time complexity of our algorithm by a factor of 2. The modification involves keeping track of old-colored clusters, and their neighborhoods separately (see [Cow93] for details.)

The second component used in Algorithm `Sparse`, namely, Procedures `Decomp`, will be bound to one of the two algorithms of [PS92, LS91]. Again, we will only state the main relevant properties of these algorithms here, and refer the interested reader to the appropriate source for more details on their structure and operation. The claims we will rely on are the following.

Lemma 4.2 [PS92] There exists an algorithm $\texttt{Decomp}_{[PS]}$ that given a graph $G=(V,E)$, $|V|=n$, constructs a weak $(2^{\sqrt{\log n}}, 2^{3\sqrt{\log n}}, 1)$-decomposition for G in $2^{c\sqrt{\log n}}$ time for some constant $c>0$.

Lemma 4.3 [LS91] There exists a randomized algorithm $\texttt{Decomp}_{[LS]}$ that given a graph $G=(V,E)$, $|V|=n$, constructs a weak $(\log n, \log n, 1)$-decomposition for G in $O(\log^2 n)$ expected time.

4.1.1 Algorithm `Sparse`

Algorithm `Sparse` operates as follows. It first calls Procedure `Decomp` with the power graph G^{4kt+1}. The output of `Decomp` is a weak $(\chi_{old}, d_{old}, 1)$-decomposition ψ_{old} of G^{4kt+1}, which is a weak $(\chi_{old}, (4kt+1)d_{old}, 4kt+1)$-decomposition of G. This decomposition is used in order to speed up the construction of the neighborhood cover by performing the construction for each color of the decomposition sequentially, but for each color, invoking Procedure `Cover` *in parallel* over separate clusters.

Algorithm `Sparse`(G)

Input: A graph $G=(V,E)$, $|V|=n$, and integer $k\geq 1$.
Output: A sparse $(k,t,kn^{1/k})$-neighborhood cover $\mathcal{T}$ of G.

Compute G^{4kt+1}, and invoke $\psi_{old} \leftarrow$ `Decomp`(G^{4kt+1}). /* ψ_{old} is a weak $(\chi_{old}, d_{old}, 1)$-decomposition

```
                    of $G^{4kt+1}$, or a weak $(\chi_{old}, (4kt+1)d_{old}, 4kt+1)$-decomposition of $G$.
*/
$\mathcal{T} \leftarrow \emptyset$.                /* $\mathcal{T}$ will be the new cover. */
$P \leftarrow V$;                  /* $P$ is the set of unprocessed vertices. */
For $i = 1$ to $kn^{1/k}$ do (sequentially):       /* find a $kn^{1/k}$-degree cover of $G$. */

  1. $W \leftarrow P$.
  2. For $j = 1$ to $\chi_{old}$ do (sequentially):    /* Cycle through old-color classes. */
     In parallel for each cluster $C$ colored $\psi_{old}(C) = j$ do:
       (a) Elect a leader for cluster $C$.
       (b) The leader learns the set $R = N_{2kt}(C) \cap W$.
       (c) The leader executes locally $(DR, \mathcal{DU}) \leftarrow$ Cover($R$).
       (d) Set $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{DU}$.
       (e) Set $P \leftarrow P - DR$.
       (f) Set $W \leftarrow W - R$.
```

4.2 Correctness and Analysis

Let us first argue the correctness of the resulting algorithm `Sparse`.

Fix some main iteration i of Algorithm `Sparse`, and let P_i denote the collection P at the beginning of iteration i. Consider the operation of the algorithm on cluster C of old color j. Let $DR(j, C)$ and $\mathcal{DU}(j, C)$ denote the two sets generated for this cluster in step (2c) of the algorithm.

Lemma 4.4 The collections $DR(j, C)$ and $\mathcal{DU}(j, C)$ satisfy the following properties:

(1) The t-neighborhood of every vertex in $DR(j, C)$ is contained in some cluster of $\mathcal{DU}(j, C)$.

(2) Every two clusters $T \in \mathcal{DU}(j, C)$ and $T' \in \mathcal{DU}(j', C')$ are disjoint.

(3) $|\bigcup_{j,C} DR(j, C)| \geq |P_i|^{1-1/k}$, and

(4) The (strong) diameter of $\mathcal{DU}$ clusters satisfies $\max_{T \in \mathcal{DU}(j,C)} Diam(T) \leq (2k-1) \cdot 2t$.

Proof Claims (1) and (4) follow directly from Lemma 4.1.

Claim (2) is analyzed by cases. The case where $j' = j$ and $C' = C$ also follows from Lemma 4.1 directly. For the case $j' = j$ but $C' \neq C$, the claim follows from the fact that C and C' were $(4kt+1)$-separated by construction, and by property (4) of Lemma 4.1. For the case $j' > j$ and $C' \neq C$, the claim follows from the fact that at the end of internal iteration j, all the nodes of R are eliminated from W, so they won't participate in internal iteration j'.

Claim (3) is proved similarly to Lemma 3.3. □

From Claim (3) of the last lemma we get

Corollary 4.5 After $kn^{1/k}$ (main) iterations of Algorithm `Sparse`, the set P becomes empty. □

Lemma 4.6 The output of Algorithm `Sparse` is a $(k, t, kn^{1/k})$-neighborhood cover of G.

Proof The fact that all neighborhoods $N_t(v)$ are covered by the constructed cover $\mathcal{T}$ is guaranteed by Property (1) of Lemma 4.1, combined with the fact that a vertex v is eliminated from the set P in Algorithm `Sparse` only as the result of joining the set DR following some invocation of Procedure `Cover`.

The (strong) diameter bound follows from Claim (4) of Lemma 4.4.

Finally, the degree bound follows from the fact that the algorithm loops through $kn^{1/k}$ main iterations, and each of those produces a set of disjoint clusters. □

Let us now turn to the complexity of Algorithm `Sparse`. The invokation of Procedure `Decomp` on the power graph G^{4kt+1} will yield an $O(kt)$ blowup in the running time of `Decomp`, say τ. Hence this stage will require time $O(kt\tau)$.

Once `Decomp` is called, the remaining operations involve the nested loops (with a total of $kn^{1/k} \times \chi_{old}$ internal iterations) each requiring the traversal of $O(ktd_{old})$-neighborhoods. Hence the running time for Algorithm `Sparse` is $O(d_{old}\chi_{old}k^2tn^{1/k})$. Then, in sum, `Sparse` is able to obtain a sparse t-neighborhood cover in the original graph G in time $O(kt\tau + td_{old}\chi_{old}k^2n^{1/k})$. (Recall that, for the applications, we typically set $k = \log n$, yielding time $O(t\tau \log n + td_{old}\chi_{old} \log^2 n)$.)

Calling algorithm `Sparse` with `Decomp` bound to the network decomposition algorithm $\texttt{Decomp}_{[PS]}$ of [PS92] gives the following theorem, relying on Lemma 4.2:

Theorem 4.7 There is a deterministic distributed algorithm that given a graph $G = (V, E)$, $|V| = n$, and integers $k, t \geq 1$, constructs a $(k, t, kn^{1/k})$-neighborhood cover of G in $t2^{c\sqrt{\log n}} +$

$t2^{4\sqrt{\log n}}n^{1/k}$ time for some constant $c > 0$, where each vertex is in at most $\Delta = O(kn^{1/k})$ clusters, and the maximum strong cluster diameter is $Diam(S_i) = O(kt)$.

Corollary 4.8 There is a deterministic distributed algorithm that given a graph $G = (V, E)$, $|V| = n$, and integers $t \geq 1$, constructs a $(\log n, \log n, t)$-neighborhood cover of G in $O(t2^{c\sqrt{\log n}})$ time for some constant $c > 0$, where each vertex is in at most $\Delta = O(\log n)$ clusters, and the maximum strong cluster diameter is $Diam(S_i) = O(t \log n)$.

Calling algorithm `Sparse` with `Decomp` bound to the randomized network decomposition algorithm $\texttt{Decomp}_{[LS]}$ of [LS91] gives the following theorem, relying on Lemma 4.3:

Theorem 4.9 There is a randomized distributed algorithm that given a graph $G = (V, E)$, $|V| = n$, and integers $k, t \geq 1$, constructs a $(k, t, kn^{1/k})$-neighborhood cover of G in $tO(k^2 \cdot \log^2 n \cdot n^{1/k})$ time, where each vertex is in at most $\Delta = O(kn^{1/k})$ clusters, and the maximum strong cluster diameter is $Diam(S_i) = O(kt)$.

Corollary 4.10 There is a randomized distributed algorithm that given a graph $G = (V, E)$, $|V| = n$, constructs a $(\log n, \log n, 1)$-neighborhood cover of G in $O(\log^4 n)$ time, and a $(\log n, t \log n, t)$-neighborhood cover of G in $O(t \log^4 n)$ time.

We remark that since sparse neighborhood cover algorithms can be translated into strong network decompositions of comparable parameters (cf. [AP90b, Cow93]), all complexity bounds hold for the constructions of strong network decompositions as well.

Acknowledgment

Thanks to Tom Leighton for helpful discussions.

References

[AAG87] Yehuda Afek, Baruch Awerbuch, and Eli Gafni. Applying static network protocols to dynamic networks. In *Proc. 28th IEEE Symp. on Found. of Comp. Science*, pages 358–370, October 1987.

[ABCP92] Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Low-diameter graph decomposition is in NC. In *Proc. 3'rd Scandinavian Workshop on Algorithm Theory*, pages 83–93, July 1992.

[ABCP93] Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Near-linear cost constructions of neighborhood covers in sequential and distributed environments and their applications. In *Proc. 34rd IEEE Symp. on Found. of Comp. Science*, pages 638–647. IEEE, November 1993.

[ABP91] Baruch Awerbuch, Alan Baratz, and David Peleg. Efficient broadcast and light-weight spanners. Unpublished manuscript, November 1991.

[ACG$^+$90] Baruch Awerbuch, Israel Cidon, Inder Gopal, Marc Kaplan, and Shay Kutten. Distributed control for PARIS. In *Proc. 9th ACM Symp. on Principles of Distrib. Computing*, pages 145–160, 1990.

[AGLP89] Baruch Awerbuch, Andrew Goldberg, Michael Luby, and Serge Plotkin. Network decomposition and locality in distributed computation. In *Proc. 30th IEEE Symp. on Found. of Comp. Science*, May 1989.

[AKP91] Baruch Awerbuch, Shay Kutten, and David Peleg. On buffer-economical store-and-forward deadlock prevention. In *Proc. of the 1991 INFOCOM*, 1991.

[AKP92] Baruch Awerbuch, Shay Kutten, and David Peleg. Online load balancing in a distributed network. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 571–580, 1992.

[AP90a] Baruch Awerbuch and David Peleg. Network synchronization with polylogarithmic overhead. In *Proc. 31st IEEE Symp. on Found. of Comp. Science*, pages 514–522, 1990.

[AP90b] Baruch Awerbuch and David Peleg. Sparse partitions. In *Proc. 31st IEEE Symp. on Found. of Comp. Science*, pages 503–513, 1990.

[AP92] B. Awerbuch and D. Peleg. Routing with polynomial communication-space trade-off. *SIAM J. Disc. Math*, 5(2):151–162, 1992.

[APPS92] Baruch Awerbuch, Boaz Patt, David Peleg, and Mike Saks. Adapting to asynchronous dynamic networks with polylogarithmic overhead. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 557–570, 1992.

[AR91] Y. Afek and M. Riklin. Sparser: A paradigm for running distributed algorithms. *J. of Algorithms*, 1991. Accepted for publication.

[BFR92] Yair Bartal, Amos Fiat, and Yuval Rabani. Competitive algorithms for distributed data management. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 39–50, 1992.

[Coh93] Edith Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. In *Proc. 34rd IEEE Symp. on Found. of Comp. Science*. IEEE, November 1993. to appear.

[Cow93] Lenore Cowen. *On Local Representation of Graphs and Networks*. PhD thesis, MIT, Lab. for Comp. Science, 1993.

[Lin87] Nathan Linial. Locality as an obstacle to distributed computing. In 27^{th} *Annual Symposium on Foundations of Computer Science*. IEEE, October 1987.

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

[Pel89] David Peleg. Distance-preserving distributed directories and efficient routing schemes. unpublished manuscript, 1989.

[Pel93] D. Peleg. Distance-dependent distributed directories. *Info. and Computation*, 1993. Also in Tech. Report CS89-10, The Weizmann Institute, May 89.

[PS92] Alessandro Panconesi and Aravind Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 581–592, 1992.

[Rao92] Satish Rao. Finding small edge cuts in planar graphs. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 229–240, 1992.