Key words. Distributed algorithm, Graph theory, Low tree-depth decomposition

AMS subject classifications. 68W15, 68R10, 05C85, 05C15

A DISTRIBUTED LOW TREE-DEPTH DECOMPOSITION ALGORITHM FOR BOUNDED EXPANSION CLASSES *

J. NEŠETŘIL† AND P. OSSONA DE MENDEZ‡

Abstract. We study the distributed low tree-depth decomposition problem for graphs restricted to a bounded expansion class. Low tree-depth decomposition have been introduced in 2006 and have found quite a few applications. For example it yields a linear-time model checking algorithm for graphs in a bounded expansion class. Recall that bounded expansion classes cover classes of graphs of bounded degree, of planar graphs, of graphs of bounded genus, of graphs of bounded treewidth, of graphs that exclude a fixed minor, and many other graphs.

There is a sequential algorithm to compute low tree-depth decomposition (with bounded number of colors) in linear time. In this paper, we give the first efficient distributed algorithm for this problem. As it is usual for a symmetry breaking problem, we consider a synchronous model, and as we are interested in a deterministic algorithm, we use the usual assumption that each vertex has a distinct identity number. We consider the distributed message-passing $\mathcal{CONGEST}_{\mathrm{BC}}$ model, in which messages have logarithmic length and only local broadcast are allowed. In this model, we present a logarithmic time distributed algorithm for computing a low tree-depth decomposition of graphs in a fixed bounded expansion class. In the sequential centralized case low tree-depth decomposition linear time algorithm are used as a core procedure in several non-trivial linear time algorithms. We believe that, similarly, low tree-depth decomposition could be at the heart of several non-trivial logarithmic time algorithms.

1. Introduction and Previous Work. The coloring problems present some of the key problems of combinatorial algorithms, both in theory and applications (such as scheduling). For sparse classes of structures, a very strong variant of the problem — low tree-detph decomposition — was recently isolated and this immediately found many applications to diverse algorithmic problems (for instance, graph problems that have finite integer index (FII) have linear kernels on bounded expansion classes when parametrized by the size of a modulator to constant tree-depth graphs [8]).

What are sparse classes of structures? Obvious examples include trees, bounded degree graphs, geometrically defined graphs (such as planar graphs). All these classes are generalized by the notion of *bounded expansion class* [17, 20]. This notion is defined in Section 3, in terms of density of shallow minors. Figure 1.1 displays some of the frequently studied classes which fall into the category of classes with bounded expansion. We shall see in Section 3 that classes with bounded expansion may be alternatively defined by means of low tree-depth decomposition.

Low tree-depth decomposition generalize graph coloring [19], and is the core of several linear-time algorithms for classes with bounded expansion [18, 20, 21, 22], including linear-time model checking [6, 7, 10, 13] (to be introduced in §3). As the coloring problems are one of the most central and most intensively studied problems in Distributed Algorithms [1, 9, 14, 16, 26], it is then natural to ask whether an efficient distributed algorithm exists to compute low tree-depth decomposition.

*Supported by grant ERCCZ LL-1201 of the Czech Ministry of Education and LEA STRUCO

† Computer Science Institute of Charles University (IUUK and ITI) Malostranské nám.25, 11800 Praha 1, Czech Republic (nesetril@iuuk.mff.cuni.cz). Supported by grant CE-ITI P202/12/G061 of GAČR

‡ Centre d'Analyse et de Mathématiques Sociales (CNRS, UMR 8557) 190-198 avenue de France, 75013 Paris, France and Computer Science Institute of Charles University (IUUK) Malostranské nám.25, 11800 Praha 1, Czech Republic (pom@ehess.fr). Supported by ANR STINT

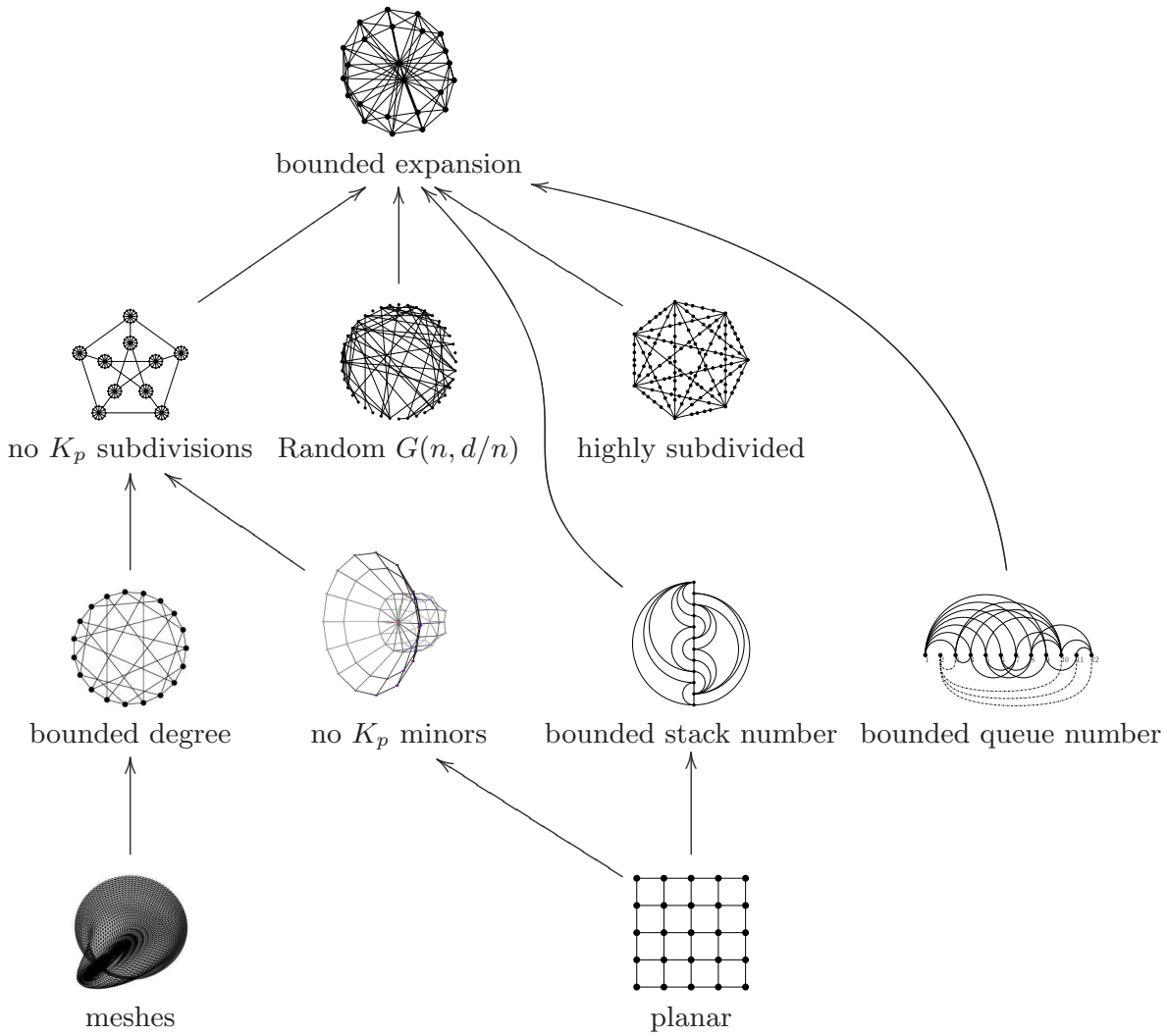

FIGURE 1.1. *Some classes with Bounded Expansion. For random graphs, the meaning of the inclusion is as follows: for every positive real $d > 0$ there exists a class $\mathcal{R}_d$ with bounded expansion, such that a random graph G with n vertices and edge probability d/n belongs to $\mathcal{R}_d$ asymptotically almost surely (as $n \to \infty$).*

The combinatorial side is natural and easy to describe. For a graph G, we denote by $|G|$ and $\|G\|$ the *order* (i.e. the number of vertices) and the *size* (i.e. the number of edges) of G. The *tree-depth* of a graph G is the minimum height of a rooted forest F such that every edge of G connects vertices that have an ancestor-descendant relationship in F [19] (see Fig. 1.2). This parameter is minor-monotone. Equivalent notions include the vertex ranking number, the minimum height of an elimination tree [4, 3, 25], etc. It is also closely related to the cycle rank of directed graphs, the star height of regular languages, and the quantifier rank of formulas. For instance, the tree-depth of a graph G with connected components $G_1, \ldots, G_k$ can be inductively defined by

$$\mathrm{td}(G) = \begin{cases} 1 & \text{if } G \text{ is } K_1 \\ \max_i \mathrm{td}(G_i) & \text{if } G \text{ is not connected} \\ 1 + \min_{v \in V(G)} \mathrm{td}(G - v) & \text{otherwise} \end{cases}$$

For non-negative integer p, a *low tree-depth decomposition* with parameter p of a graph G is a partition $V_1, \ldots, V_k$ of its vertex set such that every $i \leq p$ parts induce a

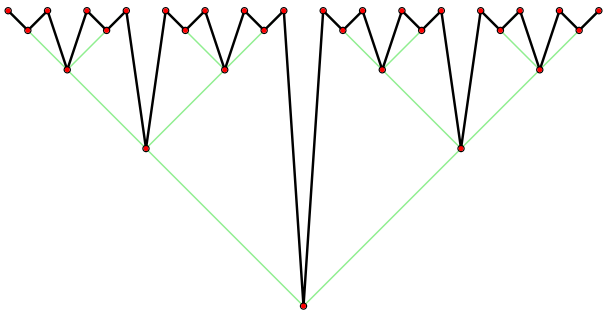

FIGURE 1.2. *The tree-depth of a path P_{2^n-2} is n.*

subgraph with tree-depth at most i [19]. The minimum number of parts k for which a low tree-depth decomposition with parameter p of G exists is $\chi_p(G)$. In particular, $\chi_1(G)$ is the standard chromatic number $\chi(G)$ of the graph G.

A related notion is the notion of $(p+1)$-*centered coloring*, which is a vertex coloring such that, for any (induced) connected subgraph H, either some color appears exactly once in H, or H gets at least $p+1$ colors (see Fig. 1.3). Every $(p+1)$-centered coloring is a low tree-depth decomposition with parameter p. Indeed, assume $V_1, \ldots, V_k$ are the color classes of a $(p+1)$-centered coloring of the graph G, and let $I \subset \{1, \ldots, k\}$ be a subset of at most p indices. Let us prove by induction over $|I|$ that the tree-depth of the subgraph G_I of G induced by $\bigcup_{i\in I} V_k$ is at most I: if $|I| = 1$, then G_I has tree-depth 1 as each V_i is an independent set. Assume that the property has been proved for subsets of at most $1 \leq t < p$ indices at let $|I| = t+1$. Let H be a connected component of G_I. As H gets less than p colors, some color i_0 appears exactly once in H at a vertex v_0. Thus

$$\mathrm{td}(H) \leq 1 + \mathrm{td}(H - v_0) \leq \mathrm{td}(G_{I\setminus\{i_0\}}) \leq 1 + |I \setminus \{i_0\}| = |I|.$$

(Note, however, that a low tree-depth decomposition with parameter p does not directly define a $(p+1)$-centered coloring, as witnessed by the path P_6 colored $1, 2, 3, 1, 2, 3$, which defines a low tree-depth decomposition of depth 3 of P_6, but is not a 4-centered coloring of P_6.)

Let us now precise the distributed computing models we shall consider. Distributed algorithms are commonly analyzed in either the $\mathcal{LOCAL}$ or the $\mathcal{CONGEST}$ model. In both settings, the distributed system is a communication network, is a graph $G = (V, E)$ of order n, where each vertex gets a unique label Id, called *identifier*, of length $\log n$ (usually assumed to be $1, \ldots, n$). The graph does not change during the execution of an algorithm. The processors in the network are located at the vertices of G, while edges of G are communication links between the processors located at the incident vertices. The vertices perform arbitrary computations in discrete synchronous rounds and, in each round, exchange messages along the edges. Precisely, in each round each vertex v is allowed to send

(i) possibly distinct messages of arbitrary size to all or part of its neighbors, in the $\mathcal{LOCAL}$ model;

(ii) possibly distinct messages of size $O(\log n)$ to all or part of its neighbors, in the $\mathcal{CONGEST}$ model;

(iii) a single message (possibly distinct for different v) of size $O(\log n)$ to all its neighbors (*local broadcast*), in the $\mathcal{CONGEST}_{\mathrm{BC}}$ model.

All messages that are sent in a round arrive before the next round starts. The number of rounds required by the computation is called the *running time* of the algorithm. For

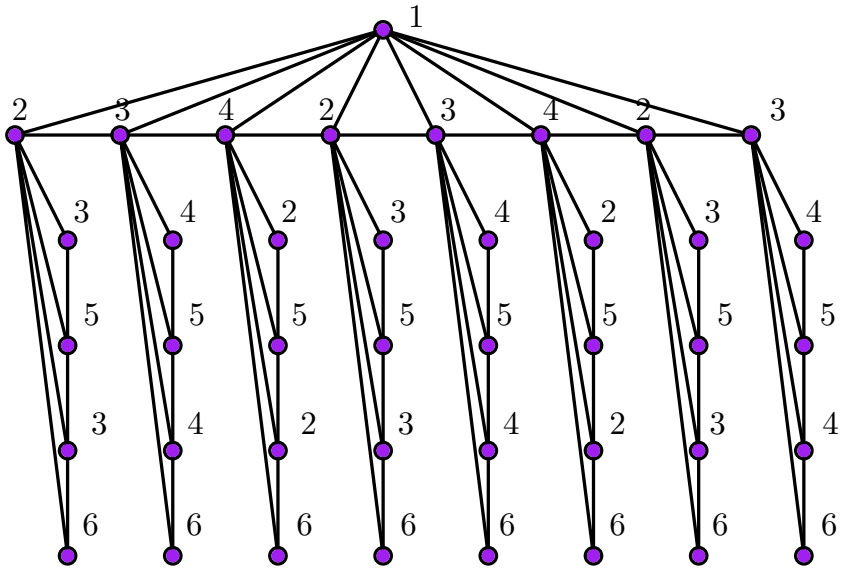

FIGURE 1.3. *Example of a* 3*-centered coloring: every connected subgraph with less than* 3 *color contains a uniquely colored subgraph. Hence this defines a low-tree depth decomposition with parameter* 2*: each color class induces a subgraph with tree-depth* 1 *(that is: a discrete graph), and every two color classes induce a subgraph with tree-depth* 2 *(that is: a star forest).*

further details on the distributed message-passing $\mathcal{LOCAL}$ and $\mathcal{CONGEST}$ models, we refer the reader to [24]. Compare also [12] for an interesting general setting.

2. Statement of the Result and Organization of the Paper. In this paper, we consider $\mathcal{CONGEST}_{\mathrm{BC}}$ model, with the additional restriction that the local data space at a vertex v is bounded by $O(\deg(v)\log n)$, where $\deg(v)$ denotes the degree of the vertex v in the network. (This means intuitively that the more connected is a vertex, the more local storage it can have.) Moreover, as in [1], we further assume that every vertex knows the order n of the graph, or at least some polynomial estimate of it. All these assumptions are natural in our setting of the problem.

We prove that, in this model, low tree-decomposition (with number of colors $N(\mathcal{C})$) can be computed in $O(\log n)$ time for graphs in a fixed bounded expansion class $\mathcal{C}$ (see Section 3 for a formal definition of bounded expansion classes). The corresponding algorithm in Section 4 as Algorithm 2. This may seem to be surprising at first glance in view of the generality of bounded expansion classes. Recall that bounded expansion classes include, for instance, proper minor closed classes like planar graphs, classes of graphs with bounded degrees, several geometrically defined classes of graphs, etc. [23]. Precisely, we prove the following.

THEOREM 2.1. *For a fixed bounded expansion class* $\mathcal{C}$*, Algorithm 2 computes (in the* $\mathcal{CONGEST}_{\mathrm{BC}}$ *model) on a graph* $G \in \mathcal{C}$*, for an input parameter* p*, a* $(p+1)$*-centered coloring (hence a low tree-depth decomposition of depth* p*) with* $O(1)$ *colors in time* $O(\log n)$*.*

In Section 3 we introduce necessary preliminaries on bounded expansion classes and fraternal augmentation. In Section 4 we discuss how the sequential algorithm Algorithm 1 can be transformed into distributed Algorithm 2, and the four main procedures involved (bounded indegree orientation, fraternal augmentation, transitive augmentation, and final coloring computation). Section 5 is devoted to the analysis of the algorithm characteristics. How the rooted trees witnessing tree-depth at most p

of a subgraph induced by p colors of the computed decomposition can be computed in a distributed way is discussed in Section 6. Finally, some final remarks and discussion are the subject of concluding Section 7.

3. Preliminaries. The graph invariants χ_p (defined by means of low tree-depth decomposition) are related to the densities of the shallow minors (topological minors, or immersions) of the graph G [20]. In order to explicit this non-trivial connection, we briefly introduce some notions related to shallow minor densities:

For a graph G and a half positive integer p we denote by $G \widetilde{\nabla} p$ the set of all the graphs H such that some subdivision of H in which every edge has been replaced by a path of length at most $2p+1$ (that is some $\leq 2p$-*subdivision* of H) is a subgraph of G. A graph $H \in G \widetilde{\nabla} p$ is called a *shallow topological minor* of G at *depth* p. In particular, $G \widetilde{\nabla} 0$ is the set of all the subgraphs of G. Let $\mathcal{C}$ be a class of graphs and let $p \in \mathbb{N}$. We extend the above notation as follows:

$$\mathcal{C} \widetilde{\nabla} p = \bigcup_{G \in \mathcal{C}} G \widetilde{\nabla} p.$$

The class $\mathcal{C}$ has *bounded expansion* if there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that it holds

$$\forall p \in \mathbb{N},\ \forall G \in \mathcal{C} \widetilde{\nabla} p, \quad \frac{\|G\|}{|G|} \leq f(p).$$

Defining

$$\widetilde{\nabla}_p(G) = \max_{H \in G \widetilde{\nabla} p} \frac{\|H\|}{|H|}$$

this turns into

$$\forall p \in \mathbb{N},\ \forall G \in \mathcal{C}, \quad \widetilde{\nabla}_p(G) \leq f(p).$$

For an extensive study of bounded expansion classes and of low tree-depth decomposition, we refer the reader to [22]. The main property of classes with bounded expansion relates to χ_p invariants.

THEOREM 3.1 ([20]). *A class $\mathcal{C}$ has bounded expansion if and only if for every $p \in \mathbb{N}$ it holds*

$$\sup_{G \in \mathcal{C}} \chi_p(G) < \infty$$

Theorem 3.1 is perhaps most useful in its algorithmic version [21], which states that, within a fixed bounded expansion class $\mathcal{C}$ and for any fixed $p \in \mathbb{N}$, a $(p+1)$-centered coloring (hence a low tree-depth decomposition with parameter p) of an input graph $G \in \mathcal{C}$ using $N(\mathcal{C})$ colors can be computed in linear time. In this paper, we will follow the lines of a slightly different linear time algorithm introduced in [22]:

THEOREM 3.2 ([22], Theorem 17.1). *For every integer p there exists a polynomial P_p (of degree about 2^{2^p}) such that for every graph G Algorithm 1 computes a $(p+1)$-centered coloring of G with $N_p(G) \leq P_p(\widetilde{\nabla}_{2^{p-2}+\frac{1}{2}}(G))$ colors in time $O(N_p(G)n)$-time.*

Algorithm 1, which is mentioned in Theorem 3.2 and described below, is based on the notion of "fraternal augmentation" (see Fig 3.1). The idea behind the use of

fraternal augmentation comes from a tree experience. Note that any vertex in the closure of a tree of height d (oriented from the root) has indegree bounded by d, and that the in-neighbors of any vertex induce a complete graph. A fraternal augmentation of a directed graph mimics this by adding at each step edges between in-neighbors of vertices, keeping track of the "depth" of the added edges. The added edges are then oriented, while keeping the maximum indegree (relatively) small. Repeating this process exponentially many times and then computing transitivity edges at depth p — that is all the pairs (x, y) of non-adjacent vertices such that there exists a directed path from x to y with length at most p — is then enough to ensure that any proper coloring of the augmented graph is automatically a $(p + 1)$-centered coloring of the (non-augmented) original graph. The reason for the exponential bound in the number of iterations is that the longest path a graph of tree-depth t can contain has length $2^n - 2$.

The fraternal augmentation procedure will be needed in the sequel, and it may be described as follows (see also [22], §7.4):

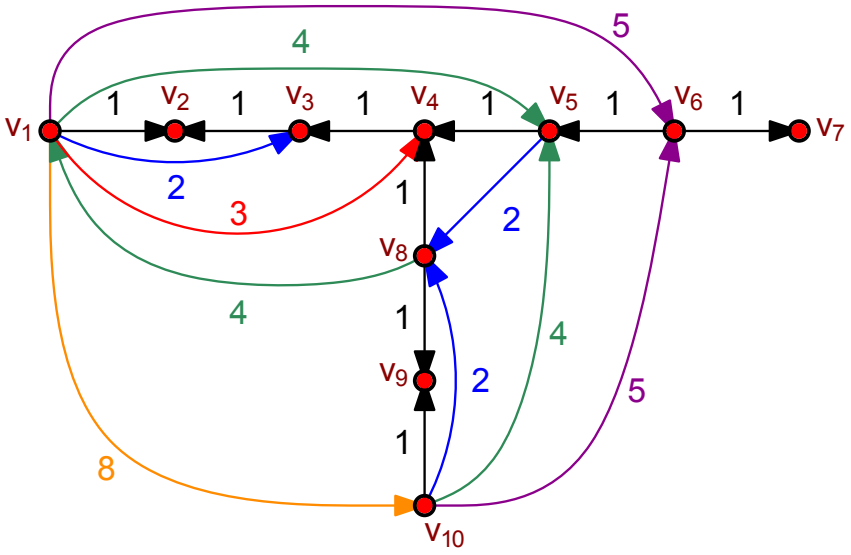

FIGURE 3.1. *Example of a fraternal augmentation*

DEFINITION 3.3. *Let $p \in \mathbb{N}$ and let $G = (V, E)$ be a graph. A* depth-p fraternal augmentation *of G is a directed graph $\vec{G}_p = (V, F)$, with edge set F partitioned as $E_1 \cup \cdots \cup E_p$, and such that*

(i) *(V, E_1) is an orientation of (V, E);*
(ii) *$(x, y) \in F$ implies $(y, x) \notin F$;*
(iii) *for every $1 \le i \le j \le p$ with $i + j \le p$, and for every $x, y, z \in V$, it holds*

$$(x, z) \in E_i \text{ and } (y, z) \in E_j \implies (x, y) \text{ or } (y, x) \text{ belongs to } \bigcup_{k=1}^{i+j} E_k.$$

DEFINITION 3.4. *Let $p \in \mathbb{N}$ and let $\vec{G}$ be a directed graph. A* depth p transitivity edge *of $\vec{G}$ is a the pair (x, y) of non-adjacent vertices such that there exists a directed path from x to y with length at most p.*

The depth p transitive augmentation *of $\vec{G}$ is the addition to $\vec{G}$ of all the depth p transitivity edges of $\vec{G}$.*

Using these notions we can give a high-level description of the sequential algorithm, presented bellow as Algorithm 1.

Algorithm 1: Sequential $(p+1)$-centered coloring algorithm

As input this algorithm receives an undirected graph $G = (V, E)$ and a parameter p, and the algorithm outputs a $(p+1)$-centered coloring of G. To do so, it first computes a depth $2^{p-1}+2$ fraternal augmentation of G, by iteratively adding depth i edges $(2 \leq i \leq 2^{p-1}+2)$ as follows:

- the edges of E_{i-1} (with E_1 defined as the original edge set E of G) are oriented with maximum indegree at most $2\widetilde{\nabla}_0(H_i)$, where $H_i = (V, E_i)$ (hence $H_1 = G$, and H_i only contains edges added during the previous iteration if $i > 1$);
- the edge set E_i is defined as the set of all pairs $\{x, y\}$ not in $\bigcup_{j=1}^{i-1} E_j$ such that there exists a vertex z with $(x, z) \in E_k$ and $(y, z) \in E_{i-k}$ for some $1 \leq k < i$.

From the graph $G^+ = (V, \bigcup_{i=1}^{2^{p-1}+2} E_i)$ we construct a directed graph G_0 obtained by adding all the depth p transitivity edges, that is all the pairs (x, y) of non-adjacent vertices such that there exists in G^+ a directed path from x to y with length at most p. Then we compute a proper vertex coloring of G_0 by using the greedy coloring algorithm (driven by topological sort ordering). The computed coloring of V (which uses at most $2\widetilde{\nabla}_0(G_0)+1$ colors) is a $(p+1)$-centered coloring of G.

4. The Distributed Algorithm. In order to design a distributed version of Algorithm 1, several changes have to be considered:

- The basic orientation (and coloring) procedures used in Algorithm 1 are based on topological sort. In the context of distributed computing, we shall rely on a variant of the algorithm of Barenboim and Elkin [1]; the orientation procedure orient will be described in more details in §4.1, while the coloring procedure color will be described in §4.4. The procedure orient will be given a constant C_i (when orienting edge set E_i) corresponding to an upper bound for the maximum indegree of the computed orientation.
- As the graph topology cannot be changed in $\mathcal{CONGEST}_{\mathrm{BC}}$ model, we cannot add edges to the graph. Instead, we emulate the different augmented graphs in G by appropriate $O(1)$-time procedures, relying on the locality of the emulated links and an upper bound on the subsequent congestion. The computation of the needed routing information for fraternal augmentation will be presented in §3.1, while the the broadcast procedure (tbc) in the graph G_0 will be presented in §4.3.

The main difficulty here is to justify and compute the bounds C_i for the orientations and a bound for the congestion of our routing algorithm. This will be done in Section 5.

A high level description of the distributed algorithm obtained from Algorithm1 is given below as Algorithm 2.

In order to allow a unified use of all our emulations, we introduce the general procedure $\mathrm{bc}(z, \mu)$ which is responsible for the emulation of a local broadcast on the graph (V, E_z) (for $1 \leq z \leq 2^{p-1}+1$), or G_0 (for $z = 0$), by calling appropriate procedure $\mathrm{fbc}(z, \mu)$ or $\mathrm{tbc}(\mu)$ (respectively described in §4.2 and §4.3).

4.1. Bounded Indegree Acyclic Orientation. We start with the description of procedure orient(z,C), which computes an acyclic orientation of the graph $H_z = (V, E_z)$ with indegree bounded by C.

Algorithm 2: Distributed $(p+1)$-centered coloring algorithm

This algorithm runs on an undirected graph $G = (V, E)$ in a fixed bounded expansion class $\mathcal{C}$ and is given a parameter p and computes a $(p+1)$-centered coloring of G. To do so, it first computes inductively the routing information of a depth $2^{p-1}+2$ fraternal augmentation of G as follows:

- the edges of E_{i-1} are acyclically oriented with maximum indegree at most C_i, where C_i is a constant defined from the class $\mathcal{C}$;
- the routing information for the edge set E_i is computed.

The routing information of the depth p transitivity edges is computed. From an acyclic orientation of all the edges of the augmented graph G_0 with maximum indegree C_0 (constant computed from $\mathcal{C}$), a proper vertex coloring of G_0 is coloring, which defines a $(p+1)$-centered coloring of G.

Input: z is the index of the graph to emulate, μ the message to broadcast

if $z = 0$ **then call** tbc(μ) ; `/* broadcast in` $G_0 = (V, E_0)$ `*/`
else if $z = 1$ **then broadcast** μ ; `/* broadcast in` $G = (V, E_1)$ `*/`
else call fbc(z, μ) ; `/* broadcast in` $H_z = (V, E_z), z > 1$ `*/`

Procedure bc(z,μ): Emulation of a local broadcast on (V, E_z)

This procedure computes the lists Neighbor_z (resp. Neighbor_z^-) of the identifiers of the neighbors (resp. the in-neighbors) of the vertices, as well as a partition, which can be used to compute a proper coloring of (V, E_z).

We mainly follow the lines of [1] (except that we consider the maximum average degree $2\widetilde{\nabla}_0(H)$ of the graph H instead of its arboricity, which is a minor change as both invariants are equivalent up to a factor of 2 but has some non-negligible impact on our bounds because of the cumulative effect of exponential number of iterations). The parameter $\epsilon > 0$ below is considered as fixed.

Procedure orient takes two global parameters, z and C, with following meaning:

z: this parameter is the index of the edge set to be oriented. For instance, the value $z = 1$ corresponds to edge set E_1, that is the original edge set of the graph. As the graph is not actually modified, this index will be used as an argument to the emulation procedure bc, which is responsible for the emulation of a local broadcast on the graph (V, E_z).

C: this parameter is an upper bound for $2(1+\epsilon)\widetilde{\nabla}_0(H_z)$, for the graph $H_z = (V, E_z)$. This will be a bound on the indegree of the computed orientation.

The result of the orientation procedure consists into

(i) the list Neighbor_z^- containing the identifiers of the in-neighbors of the considered vertex;

(ii) the list Neighbor_z containing the identifiers of all the neighbors of the considered vertex;

(iii) the integer Part (in $\{1, \ldots, \lfloor \log n / \log(1+\epsilon) \rfloor\}$), containing the index of the part the considered vertex belongs to, in a special partition of the vertices, which can be used by procedure color to compute a proper coloring of H_z

The procedure proceeds as follows:

(i) The list Neighbor_z, as well as the degree d of the vertex, is computed using

Input: z is the index of the graph $H_z = (V, E_z)$ to be used, C is an integer such that $C \geq 2(1+\epsilon)\widetilde{\nabla}_0(H)$. Both parameters are **global** (do not depend on the vertex running the code)

Result: Part is the index of the part V_{Part} the vertex belongs to ($1 \leq \text{Part} \leq q = \log n / \log(1+\epsilon)$) and it is such that the vertex has at most C neighbors in $\bigcup_{i \geq \text{Part}} V_i$. The list Λ (resp. Λ') contains the identifiers of the neighbors in $\bigcup_{i > \text{Part}} V_i$ (resp. the neighbors in V_{Part} with identifier $> \text{Id}$). The lists Neighbor_z (resp. Neighbor_z^-) contains the identifiers of the neighbors (resp. the in-neighbors) of the vertex.

call bc(z, Id); `// locally broadcast the identifier`
put all received identifiers in set Neighbor_z;
let d be the number of received messages;
let $\text{Part} = 0$;
let $\Lambda = \Lambda' = ()$;
let $q = \log n / \log(1+\epsilon)$;
for $i = 1$ **to** q **do**
  if $d \leq C$ **and** $\text{Part} = 0$ **then**
    let $\text{Part} = i$;
    call bc(z, Id);
  if $\text{Part} = 0$ **then**
    decrease d by the number of received messages;
  else if $\text{Part} < i$ **then**
    add all received identifiers to Λ;
  else
    add all received identifiers greater than Id to Λ';
put all identifiers in Λ and Λ' in the set Neighbor_z^-.

Procedure orient(z, C) Partition and Orientation

a local broadcast. In the context of the procedure, the value d will be the number of *unmarked* neighbors of a vertex.

(ii) partition is computed by iteratively marking unmarked vertices with at most C unmarked neighbors, putting newly marked vertices into a new part, and updating accordingly the value d of the number of unmarked neighbors. That $q = \lfloor \log n / \log(1+\epsilon) \rfloor$ iterations are sufficient is proved in Lemma 4.1 below. By construction a vertex in part V_{Part} has at most C neighbors in $\bigcup_{i \geq \text{Part}} V_i$.

(iii) The list Neighbor_z^- is computed, by considering the acyclic orientation defined by lexicographic order of (Part, Id).

We shall now prove that the procedure orient(z, C) indeed computes in $O(\log n)$-time an acyclic orientation of the graph H_z with indegree at most C.

LEMMA 4.1. *Procedure orient(z, C) computes in $O(\log n)$-time an acyclic orientation of the graph $H_z = (V, E_z)$ with indegree at most C.*

At the end of the procedure, the list Neighbor_z *(resp.* Neighbor_z^-*) of a vertex v contains the identifiers of all the neighbors (resp. in-neighbors) of v.*

Proof. Let us first prove that at the end of Procedure orient(z, C), each vertex v has a been assigned a value $\text{Part} \neq 0$. For $i \geq 0$, let n_i be the number of vertices with $\text{Part} = 0$ before step i. As the subgraph G_i induced by vertices with $\text{Part} = 0$ has average degree at most $C/(1+\epsilon)$, the number n_{i+1} of vertices having degree greater

than C is at most $n_i/(1+\epsilon)$ (by Markov inequality). Hence after $q = \log n/\log(1+\epsilon)$ steps every vertex has Part $\neq 0$.

It follows that all the edges of G are eventually oriented by Procedure orient(z, C). By construction, the computed orientation is acyclic and has indegree at most C. □

4.2. The Fraternal Augmentation Emulation. In this section, we describe how to emulate a local broadcast in the graph (G, E_z), defined as follows: two vertices $u, v \in V$ are adjacent in E_z if

1. vertices u and v are adjacent in none of the (V, E_i) with $1 \leq i < z$;
2. there exists a vertex w and integers $1 \leq i, j < z$ such that $(u, w) \in E_i$, $(v, w) \in E_j$, and $i + j = z$.

Procedure fbc(z, μ) allows vertices to broadcast to all their neighbors a message μ (possibly different for all sending vertices). The procedure fbc(z, μ) is inductive in nature: in order to In order to emulate a broadcast in H_z, it uses broadcasts in graphs H_i (for $i < z$) and even needs these graphs to have been oriented (with maximum indegree C_i), and neighbor and in-neighbor list Neighbor$_i$ and Neighbor$_i^-$ to have been computed. Recall that the edges in H_z corresponds to pairs of vertices $\{u, v\}$ such that

- the pair $\{u, v\}$ is not in any E_i for $i < z$;
- there exists at least a vertex w and an integer $1 \leq d \leq z - 1$ such that $(u, w) \in E_d$ and $(v, w) \in E_{z-d}$.

The method used by the broadcast fbc(z, μ) then follows. To get an intuitive view on how the algorithm proceeds, consider how a broadcast from vertex u will reach a neighbor v of u in H_z. Note that, as mentioned above, there exists at least a vertex w and an integer $1 \leq d_0 \leq z - 1$ such that $(u, w) \in E_{d_0}$ and $(v, w) \in E_{z-d_0}$. The procedure fbc(z, μ) repeats at loop for values $d = 1, \ldots, z-1$. Consider the loop when $d = d_0$. At this stage, vertices (including u) broadcast on E_d a message containing both the message μ they want to broadcast and their identifier Id. As $(u, w) \in E_d$, vertex w will receive this message. As the message comes from an in-neighbor of w in H_d (checked by verifying that the received identifier i of u belongs to the list Neighbor$_d^-$ of in-neighbors in H_d), the message (i, m) is put in the list L' of relevant messages. (Note that this list will contain at most one message per in-neighbors in H_d hence will have length at most C_d.) Then, each relevant message in L' is broadcast in E_{z-d} with addition of the identifier of the current vertex (w in our example) to the message. This message will then reach v, which will detect that the message comes from an out-neighbor in E_{z-d} (as $(v, w) \in E_{z-d}$). After having checked that the identifier of the origin vertex (here u, identified by i) is not in the neighbor list for some $i < z$, the message formed by the pair of the identifier of the origin vertex (here u) and its message μ is added to the reception list RList. Duplicates may exist in RList, as the vertex w considered above does not have to be unique for the pair (u, v). Nevertheless, as the identifier of the origin is known, it is easy to remove duplicates and to fill a clean list RcvList with all the messages the vertex has received from different origins. This is formalized as procedure fbc(z, μ) bellow.

In Procedure fbc(z, μ), the graph (V, E_z) is only implicitly defined using graphs (V, E_i) $(1 \leq i < z)$. The neighbor list Neighbor$_z$ is not computed by this procedure, but it can computed afterward by the orientation procedure orient(z,C_z), which uses Procedure bc(z, μ).

Remark 1. *Local data space at vertex v is $O(\deg(v) \log n)$. Indeed, it is easily checked by induction that for each vertex v of G, the ratio of the degree of v in $\vec{G}_p$*

Require: for $1 \leq d < z$, sets Neighbor_d and Neighbor_d^- are defined and the maximum indegree in (V, E_d) is bounded by C_d.
Input: z is the index of the emulated graph (V, E_z), μ is the message to broadcast to all neighbors in (V, E_z).
Output: RcvList contains all received messages

```
let RcvList = ();                                   // Received messages
let RList = ();                                     // Received messages with Ids
for d = 1 to z − 1 do
    call bc(d, (Id, μ));
    let L' = (). forall the received message (i, m) do
        if i ∈ Neighbor_d^- then add (i, m) to L'
    for k = 1 to C_d do
        if L' ≠ () then
            pop (i, m) from L';
            call bc(z − d, (i, Id, m));
        forall the received message (i, j, m) do
            if j ∈ Neighbor_{z−d} \ Neighbor_{z−d}^- and i ∉ ⋃_{k=1}^{z−1} Neighbor_k then
                add (i, m) to RList;
purge duplicates from RList;
forall the message (i, m) in RList do put m in RcvList
```

Procedure fbc(z,μ) Local broadcast on (V, E_z)

and the degree of v in G is bounded by a constant, which only depends on the bounded expansion class and the integer p.

That procedure $\text{fbc}(z, \mu)$ *is run in constant time easily follows from the existence of a bound C_i on the maximum indegree of (V, E_i) for $1 \leq i < z$. The existence of this bound is non-trivial and is justified in §5.*

4.3. The Transitive Augmentation. Carrying on the transformation of Algorithm 1 into Algorithm 2, we reach the step where depth p transitivity arcs have to be computed on $\vec{G}_q$. In distributed setting, this translates into a procedure $\text{tbc}(\mu)$, which is emulating the local broadcast in the augmented graph $G_0 = (V, E_0)$. This procedure is described bellow. Although similar to §4.2, this procedure needs to be formalized carefully.

Procedure $\text{tbc}(\mu)$ needs to keep on the messages an information on the direction of the broadcast ("+" for messages following forward arcs in $(V, \bigcup_{i=1}^{q} A_e)$, and "-" for messages following backward arcs). In the procedure, the constant $X_{p,q}$ (bounding the number of iterations of the main loop) is computed from p and the bounds $C_1, \ldots, C_q$ on the maximum indegrees in the directed graphs $(V, E_1), \ldots, (V, E_q)$. Existence of the bounds C_i is non-trivial, and is justified in Section 5.

4.4. The Final Coloring. Procedure $\text{orient}(z, C)$ not only computes an acyclic orientation with bounded indegree, but also computes a partition of the vertices which is used by the coloring procedure color (in a similar way that in [1]). This procedure runs in $O(\log n)$-time. (Note that in Procedure color, the coloring of subgraphs $G[V_i]$, which have maximum degree at most C, is done in $O(C + \log^* n)$-time using [2].)

Require: sets $\mathrm{Neighbor}_i$ and $\mathrm{Neighbor}_i^-$ have been computed for all $1 \leq i \leq q$.
Input: μ is the message to broadcast to all G_0-neighbors.

let RcvList = ();
let RList = ();
push $(\mathrm{Id}, \mu, p, \text{“+”})$ on L';
push $(\mathrm{Id}, \mu, p, \text{“-”})$ on L';
for $i = 1$ **to** $X_{p,q}$ **do**
  if $L' \neq \emptyset$ **then**
    pop $(i, m, z, \mathrm{direction})$ from L';
    for $d = 1$ **to** q **do**
      call $\mathrm{bc}(d, (\mathrm{Id}, d, i, m, k-1, \mathrm{direction}))$;
      `// params: identifier of the sender for last hop, length of the last hop, identifier of the initial sender, main message, number of remaining hops, direction of the broadcast`
      forall the *received message* $(f, l, i', m', k', \mathrm{direction}')$ **do**
        if $k' = 0$ **then**
          add (i', m') to RList;
        else if $f \in \mathrm{Neighbor}_l^-$ **and** $\mathrm{direction}' =$ *“+”* **then**
          add $(i', m', k', \mathrm{direction}')$ to L';
        else if $f \in \mathrm{Neighbor}_l \setminus \mathrm{Neighbor}_l^-$ **and** $\mathrm{direction}' =$ *“-”* **then**
          add $(i', m', k', \mathrm{direction}')$ to L';
purge duplicates from RcvList;
forall the *message* (i, m) *in* RList **do put** m in RcvList

Procedure $\mathrm{tbc}(\mu)$ Local broadcast on G_0

Output: γ is the color of the vertex

Compute a $(C_0 + 1)$*-coloring* c *of every* $G_0[V_i]$;
let $q = \log n / \log(1 + \epsilon)$;
let $L = ()$;
for $i = 1$ **to** q **do**
  for $j = 1$ **to** $C_0 + 1$ **do**
    if $(\mathrm{Part}, c) = (i, j)$ **then**
      let γ be the smallest integer not in L;
      call $\mathrm{bc}(0, \gamma)$;
    else if $(\mathrm{Part}, c) > (i, j)$ **then**
      add all received values to L;

Procedure color (Coloring from Orientation of G_0)

5. Analysis and Main Theorem. The analysis of our algorithm is tdeious however routine as all its parts and subroutines fit well together. In the following proof, we use all the special notations introduced in Section 4. One of the main ingredients of the proof stands in the stability of the invariants $\widetilde{\nabla}_r(G)$ under lexicographic product by a small clique. Lexicographic product $G \bullet K_p$ will reflect the intuition of

a routing with congestion p (see Fig. 5.1).

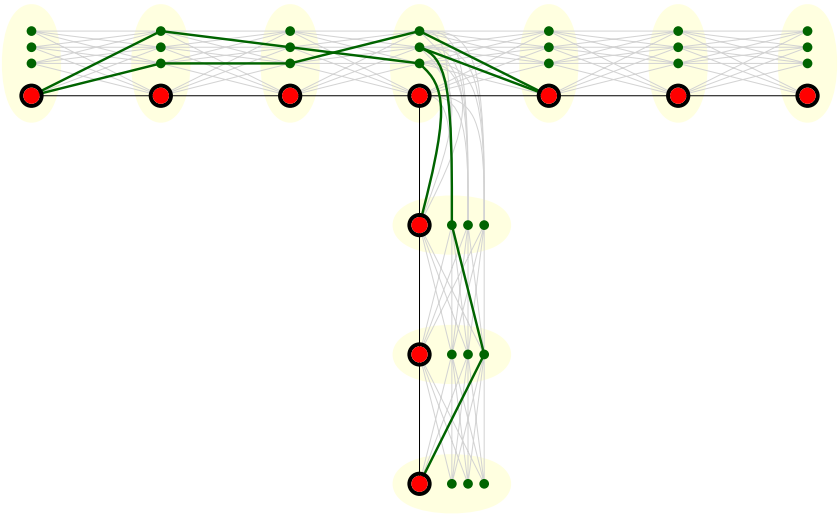

FIGURE 5.1. *Routing in the lexicographic product $G \bullet K_p$*

Recall that for graph G and integer p, the *lexicographic product* $G \bullet K_p$ of the graph G and the complete graph K_p is the graph with vertex set $V(G) \times \{1, \dots, p\}$, where (u, i) and (v, j) are adjacent if either $u = v$ or $\{u, v\}$ is an edge of G. For a proof of the following lemma, we refer the reader to [22], Proposition 4.6.

LEMMA 5.1. *Let G be a graph, let $p \geq 2$ and r be positive integers. Then*

$$\widetilde{\nabla}_r(G \bullet K_p) \leq \max(2r(p-1)+1, p^2)\, \widetilde{\nabla}_r(G) + p - 1.$$

□

Proof of Theorem 2.1. We now consider how the values C_i are fixed. In the following, we fix a bounded expansion class $\mathcal{C}$ and a positive real $\epsilon > 0$.

The constant C_1 is constrained by $C_1 \geq 2(1+\epsilon)\widetilde{\nabla}_0(G)$ (see §4.1). Hence we define

$$C_1 = \lceil 2(1+\epsilon) \sup_{G \in \mathcal{C}} \widetilde{\nabla}_0(G) \rceil.$$

Let $F(1) = 0$. For $2 \leq i \leq p$ define inductively

$$F(i) = \begin{cases} \binom{C_1}{2}, & \text{if } i = 2; \\ \sum_{j=2}^{i-1} F(j)C_{i-j} & \\ \quad + \sum_{j=1}^{(i-1)/2} C_j C_{i-j}, & \text{if } i \equiv 1 \pmod 2; \\ \sum_{j=2}^{i-1} F(j)C_{i-j} & \\ \quad + \sum_{j=1}^{i/2-1} C_j C_{i-j} + \binom{C_{i/2}}{2}, & \text{if } i \equiv 0 \pmod 2. \end{cases}$$

Then (see §7.4 of [22]) by subdividing $j-1$ times the arcs of $\vec{G}_i$ that are in A_j (for $1 \leq j \leq i$) we obtain a subgraph of the lexicographic product $G \bullet K_{1+F(i)}$ of G with a complete graph of order $F(i)+1$. It follows that if H_i denotes the graph (V, E_i) it holds $\widetilde{\nabla}_0(H) \leq \widetilde{\nabla}_{\frac{i-1}{2}}(G \bullet K_{1+F(i)})$. According to Lemma 5.1, in order to have the inequality $C_i \geq 2(1+\epsilon)\widetilde{\nabla}_0(H_i)$ satisfied, we can safely define

$$C_i = \left\lceil 2(1+\epsilon)\big(\max((i-1)F(i)+1, (F(i)+1)^2) \sup_{G \in \mathcal{C}} \widetilde{\nabla}_{\frac{i-1}{2}}(G) + F(i)\big) \right\rceil < \infty$$

Then we see easily that for each $1 \leq i \leq q$ the maximum indegree of (V, E_i) is bounded by C_i.

It follows that the number of colors in the $(p+1)$-centered coloring computed by Procedure color is bounded by $N = C_0 + 1$, where

$$C_0 = \lceil 2(1+\epsilon)S \rceil$$

and

$$S = \frac{\left(\sum_{i=1}^{q} C_i\right)^{p+1} - \left(\sum_{i=1}^{q} C_i\right)}{\left(\sum_{i=1}^{q} C_i\right) - 1}.$$

This is a large yet finite number, which only depends on $\mathcal{C}$. Theorem 2.1 follows. □

REMARK 2. *The parameter ϵ has the following influence: When $\epsilon \to 0$, the orientation procedure slows down (by a factor $\log(1+\epsilon)$); However, when ϵ grows, the constants C_i grow like a power of $(1+\epsilon)$, implying a proportional increase of the time used by local broadcast emulation procedures and of the bound of the number of colors used by the algorithm. It follows that a good tradeoff for ϵ could be $\epsilon \approx 2^{-2^p}$.*

6. Computing Rooted Forests. In a $(p+1)$-centered coloring of G with N colors, for each subset I of N with cardinality at most p, the subgraph G_I induced by vertices with color in I has tree-depth at most $|I|$ thus is a subgraph of a rooted forest with height at most $|I|$. When the set of colors is ordered, the rooted forest can be defined in a canonical way. The list, for each vertex of color $\gamma \in I$, of its ancestors in the canonical forest is computed in $O(1)$-time from the coloring by Procedure tree(I). Note that for every subset I of N with cardinality at most p, every vertex is adjacent in (V, E_0) to all its ancestors.

Input: I is a subset of at most p colors in $\{1, \ldots, N\}$ (global parameter)
Output: Ancestors is ordered list of ancestors identifier and color

call bc$(0, \gamma)$;
let L be the list of received colors;
if $\gamma \in I$ **then**
  add γ to L;
  let L' be the list of the colors c appearing exactly once in L;
  call bc$(0, (\mathrm{Id}, \gamma, L'))$; // $|L'| \leq p$ `hence size`$(L') = O(\log n)$
let S be the set of the received messages;
let $J = \emptyset$;
for $i = 1$ **to** p **do**
  if $\gamma \in I \setminus J$ **then**
    let c_0 be the minimum of $\bigcap_{(i'', c'', L'') \in S} L''$;
    let (i'', c'', L'') be the only triple in S with $c'' = c_0$;
    add c_0 to J;
    if $c_0 \neq \gamma$ **then add** (i'', c_0) to Ancestors

Procedure tree(I) (compute the root for subset of colors I)

From these rooted forests, several $O(1)$-time computations can be made. For instance, if F is a connected graph of order at most p, a marking of the vertices belonging to at least one induced copy of F in G can be computed in $O(1)$-time.

7. Conclusion and Further Works. We believe that low tree-decompositions, which proved to be powerful tools to design linear time sequential algorithms, can be applied in the context of distributed computing to design efficient $O(\log n)$ algorithms.

Let us give two examples:

(i) Minimum Dominating Set problem is NP-hard to solve in general, even approximately. Much work has been dedicated to sequential and distributed approximation algorithms on restricted graph classes. For instance, a distributed algorithm is given in [15] for graphs with bounded arboricity a, which achieves a factor $O(a^2)$-approximation in randomized time $O(\log n)$. Also, a deterministic sequential linear time algorithm has been given in [5], which achieves (for graphs in an arbitrary but fixed bounded expansion class) a constant factor approximation for the distance-k Minimum Dominating Set problem. The latter algorithm is, in essence, based on low tree-depth decomposition techniques. Does there exist an $O(\log n)$-time distributed algorithm achieving a constant factor approximation for the distance-k Minimum Dominating Set problem for graphs in a bounded expansion class?

(ii) Linear-time (centralized sequential) model checking algorithms have been proposed for bounded expansion classes [6, 10, 13] (and even in the more general setting of nowhere dense classes [11]), which are based on low tree-depth decompositions. On the basis of the distributed algorithm proposed in this paper, it is natural to ask whether for arbitrary (but fixed) local first-order formula ϕ, an $O(\log n)$ distributed algorithm exists (in $\mathcal{CONGEST}_{\mathrm{BC}}$ model), which would mark vertices v of the network graph G such that $G \models \phi(v)$. (Recall that a formula $\phi(x)$ is *local* if there exists an integer r such that the satisfaction of ϕ only depends on the r-neighborhood of the free variable.)

REFERENCES

[1] L. Barenboim and M. Elkin, *Sublogarithmic distributed MIS algorithm for sparse graphs using Nash-Williams decomposition*, Distributed Computing, 22 (2010), pp. 363–379.

[2] L. Barenboim, M. Elkin, and F. Kuhn, *Distributed* $(\Delta+1)$*-coloring in linear (in* Δ*) time*, SIAM Journal on Computing, 43 (2014), pp. 72–95.

[3] H. Bodlaender, J. Deogun, K. Jansen, T. Kloks, D. Kratsch, H. Müller, and Z. Tuza, *Rankings of graphs*, in Graph-Theoretic Concepts in Computer Science, vol. 903/1995 of Lecture Notes in Computer Science, Springer, 1995, pp. 292–304.

[4] J. Deogun, T. Kloks, D. Kratsch, and H. Müller, *On vertex ranking for permutation and other graphs*, in Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science, P. Enjalbert, E. Mayr, and K. Wagner, eds., vol. 775 of Lecture Notes in Computer Science, Springer, 1994, pp. 747–758.

[5] Z. Dvořák, *Constant-factor approximation of domination number in sparse graphs*, European J. Combin., 34 (2013), pp. 833–840.

[6] Z. Dvořák, D. Král', and R. Thomas, *Deciding first-order properties for sparse graphs*, in 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2010), 2010, pp. 133–142.

[7] ———, *Testing first-order properties for subclasses of sparse graphs*, Journal of the ACM, 60:5 Article 36 (2013).

[8] J. Gajarský, P. Hliněný, J. Obdržálek, S. Ordyniak, F. Reidl, P. Rossmanith, F. Sánchez Villamil, and S. Sikdar, *Kernelization using structural parameters on sparse graph classes*, in ESA 2013, 2013. accepted.

[9] A. V. Goldberg and S. A. Plotkin, *Parallel* $(\Delta+1)$*-coloring of constant-degree graphs*, Information Processing Letters, 25 (1987), pp. 241 – 245.

[10] M. Grohe and S. Kreutzer, *Methods for algorithmic meta theorems*, in Model Theoretic Methods in Finite Combinatorics, Contemporary mathematics, 2011, pp. 181–206.
[11] M. Grohe, S. Kreutzer, and S. Siebertz, *Deciding first-order properties of nowhere dense graphs*. arXiv:1311.3899 [cs.LO], November 2013.
[12] M. Herlihy, D. Kozlov, and S. Rajsbaum, *Distributed Computing Through Combinatorial Topology*, Elsevier, 2014.
[13] W. Kazana and L. Segoufin, *Enumeration of first-order queries on classes of structures with bounded expansion*, in Proceedings of the 16th International Conference on Database Theory, 2013, pp. 10–20.
[14] F. Kuhn and R. Wattenhofer, *On the complexity of distributed graph coloring*, in Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing, 2006, pp. 7–15.
[15] C. Lenzen and R. Wattenhofer, *Minimum dominating set approximation in graphs of bounded arboricity*, in Distributed Computing, N. Lynch and A. Shvartsman, eds., vol. 6343 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 510–524.
[16] N. Linial, *Locality in distributed graph algorithms*, SIAM J. on Computing, 21 (1992), pp. 193–201.
[17] J. Nešetřil and P. Ossona de Mendez, *The grad of a graph and classes with bounded expansion*, in 7th International Colloquium on Graph Theory, A. Raspaud and O. Delmas, eds., vol. 22 of Electronic Notes in Discrete Mathematics, Elsevier, 2005, pp. 101–106.
[18] ———, *Linear time low tree-width partitions and algorithmic consequences*, in STOC'06. Proceedings of the 38th Annual ACM Symposium on Theory of Computing, ACM Press, 2006, pp. 391–400.
[19] ———, *Tree depth, subgraph coloring and homomorphism bounds*, European Journal of Combinatorics, 27 (2006), pp. 1022–1041.
[20] ———, *Grad and classes with bounded expansion I. decompositions*, European Journal of Combinatorics, 29 (2008), pp. 760–776.
[21] ———, *Grad and classes with bounded expansion II. algorithmic aspects*, European Journal of Combinatorics, 29 (2008), pp. 777–791.
[22] ———, *Sparsity (Graphs, Structures, and Algorithms)*, vol. 28 of Algorithms and Combinatorics, Springer, 2012. 465 pages.
[23] J. Nešetřil, P. Ossona de Mendez, and D. Wood, *Characterizations and examples of graph classes with bounded expansion*, European Journal of Combinatorics, 33 (2012), pp. 350–373.
[24] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*, SIAM, 2000.
[25] A. Schäffer, *Optimal node ranking of trees in linear time*, Inform. Process. Lett., 33 (1989/90), pp. 91–96.
[26] M. Szegedy and S. Vishwanathan, *Locality based graph coloring*, in Proc. 25th ACM Symposium on Theory of Computing, 1993, pp. 201–207.