What Cannot Be Computed Locally![*]

Fabian Kuhn
Dept. of Computer Science
ETH Zurich
8092 Zurich, Switzerland
kuhn@inf.ethz.ch

Thomas Moscibroda
Dept. of Computer Science
ETH Zurich
8092 Zurich, Switzerland
moscitho@inf.ethz.ch

Roger Wattenhofer
Dept. of Computer Science
ETH Zurich
8092 Zurich, Switzerland
wattenhofer@inf.ethz.ch

ABSTRACT

We give time lower bounds for the distributed approximation of minimum vertex cover (MVC) and related problems such as minimum dominating set (MDS). In k communication rounds, MVC and MDS can only be approximated by factors $\Omega(n^{c/k^2}/k)$ and $\Omega(\Delta^{1/k}/k)$ for some constant c, where n and Δ denote the number of nodes and the largest degree in the graph. The number of rounds required in order to achieve a constant or even only a polylogarithmic approximation ratio is at least $\Omega(\sqrt{\log n/\log\log n})$ and $\Omega(\log\Delta/\log\log\Delta)$. By a simple reduction, the latter lower bounds also hold for the construction of maximal matchings and maximal independent sets.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*;
G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*;
G.2.2 [**Discrete Mathematics**]: Graph Theory—*network problems*

General Terms

Algorithms, Theory

Keywords

approximation hardness, distributed algorithms, dominating set, locality, lower bounds, maximal independent set, maximal matching, vertex cover

[*]The work presented in this paper was supported (in part) by the Hasler Stiftung (Berne, Switzerland) and by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

1. INTRODUCTION AND RELATED WORK

What can be computed locally? Naor and Stockmayer [18] raised this for the distributed computing community vital question over a decade ago, and though the price of locality has puzzled researchers ever since, results have been rare [15]. The importance of locality stems from the desire of achieving a global goal based on local information only. This is not only one of the key challenges when developing fast distributed algorithms, but it also improves fault-tolerance and allows detecting illegal global configurations by checking local conditions [1]. Recent advances in networking and ever-growing distributed systems drive the need for a thorough understanding of locality issues. In the present paper, we study the impact of locality in the classic *message passing* model where the distributed system is modelled as a graph: Nodes represent the processors and two nodes can communicate if and only if they share an edge in the graph.

We present lower bounds for several traditional graph theory problems, starting out with minimum vertex cover (MVC). A vertex cover for a graph $G = (V, E)$ is a subset of nodes $V' \subseteq V$ such that, for each edge $(u, v) \in E$, at least one of the two incident nodes u, v belongs to V'. Finding a vertex cover with minimum cardinality is known as the MVC problem.

In local algorithms, nodes are only allowed to communicate with their direct neighbors in G. After several rounds of communication they need to come up with a global solution, e.g. a good approximation for MVC. Intuitively MVC could be considered perfectly suited for a local algorithm: A node should be able to decide whether to join the vertex cover by communicating with its neighbors a few times. Very distant nodes seem to be superfluous for this decision. The fact that there is a simple greedy algorithm which approximates MVC within a factor 2 in the global setting, additionally raises hope for an efficient local algorithm.

To our surprise, however, there is no such algorithm. In this paper, we show that in k communication rounds, MVC can only be approximated by a factor $\Omega(n^{c/k^2}/k)$ for a constant c larger than $1/4$. This implies a running time of $\Omega(\sqrt{\log n/\log\log n})$ in order to achieve a constant or even polylogarithmic approximation ratio. When making the result dependent on the maximum degree Δ instead of the number of nodes n, the approximation ratio is $\Omega(\Delta^{1/k}/k)$ and the time needed to obtain a polylogarithmic or constant approximation is $\Omega(\log\Delta/\log\log\Delta)$. The results for MVC can be extended to minimum dominating set (MDS) and other distributed covering problems. Further, the time lower bounds for constant MVC approximations also apply for the construction of maximal matchings and maximal independent sets. Since all lower bounds hold even in the cases of unbounded messages and complete synchrony, the lower bounds are a true consequence of locality limitations, and not merely a side-effect of congestion, asynchrony, or limited message size.

Moreover, some of our lower bounds are almost tight for many cases. In k rounds, MVC can be approximated by a factor $O(\Delta^{1/k})$ [10]. In order to obtain a polylogarithmic approximation, we have to set $k = O(\log\Delta/\log\log\Delta)$, for a constant approximation, the number of rounds is $k = O(\log\Delta)$. Hence, for polylogarithmic (and worse) approximation ratios, our lower bound is tight whereas for constant approximation algorithms, it is tight up to a factor $O(\log\log\Delta)$. For the MDS problem, recent results show that in k rounds, the respective linear program can be approximated up to a factor $\Delta^{c/\sqrt{k}}$, whereas for MDS itself an approximation ratio of $\Delta^{c/\sqrt{k}}\ln\Delta$ can be achieved [9, 10]. If message size and local computations are unbounded, the MDS LP and MDS can be appoximated up to factors $O(n^{1/k})$ and $O(n^{1/k}\ln\Delta)$, respectively [10]. It is easy to verify that the lower bounds for MDS also holds for the fractional LP version. The best known algorithms for maximal matching and maximal independent set need time $O(\log n)$ [8, 17].

A pioneering and seminal lower bound by Linial [15] shows that the non-uniform $O(\log^* n)$ coloring algorithm by Cole and Vishkin [2] is asymptotically optimal for the ring. This lower bound has been cherished by researchers as a fundamental advancement in the theory of distributed algorithms. For different models of distributed computation, there is a number of other lower bounds [6, 11], most notably for the problem of constructing a minimum spanning tree (MST) of the network graph [3, 4, 16, 20]. With the exception of [3], the MST lower bounds apply to a model where message size is bounded. In [4], the lower bounds of [16, 20] are extended to approximation algorithms. To the best of our knowledge, it is the only previous lower bound on distributed hardness of approximation.

Linial's lower bound [15] is based on the *drosophila melanogaster* of distributed computing, the ring network. For the MVC problem, highly symmetric graphs such as rings often feature a straight-forward solution with constant approximation ratio. In any δ-regular graph, for example, the algorithm which includes all nodes in the vertex cover is already a 2-approximation for MVC: Each node will cover at most δ edges, the graph has $n\delta/2$ edges, and therefore at least $n/2$ nodes need to be in the minimum vertex cover. On the other extreme, asymmetric graphs often enjoy constant-time algorithms, too. In a tree, choosing all inner nodes yields a 2-approximation. The same trade-off exists for node degrees. If the maximum node degree is low (constant), we can tolerate to choose all nodes, and have by definition a good (constant) approximation. If there are nodes with high degree, the diameter of the graph is small, and a few communication rounds suffice to inform all nodes of the entire graph. What we need is a construction of a not too symmetric and not too asymmetric graph with a variety of node degrees! Not many graphs with these "non-properties" are known in distributed computing.

The proof of our lower bounds is based on the timeless indistinguishability argument [7, 12]. In k rounds of communication, a network node can only gather information about nodes which are at most k hops away and hence, only this information can be used to determine the computation's outcome. In particular, we show that after k communication rounds two neighboring nodes see exactly the same graph topology; informally speaking, both neighbors are equally qualified to join the vertex cover. However, in our example graphs, choosing the wrong neighbor will be ruinous.

The remainder of the paper is organized as follows. Section 2 describes the model of computation. The lower bound is constructed in Section 3. In Section 4, we show how our MVC lower bound can be extended to minimum dominating set (MDS), maximal matching (MM), and maximal independent set (MIS). Section 5 concludes the paper.

2. MODEL

We consider the classic *message passing* model, which consists of a point-to-point communication network, described by an undirected graph $G = (V, E)$. Nodes correspond to processors and edges represent communication channels between them. In one communication round, each node of the network graph can send an arbitrarily long message to each of its neighbors. Local computations are for free and initially, nodes have no knowledge about the network graph. They only know their own unique identifier.[1] In k communication rounds, a node v may collect the IDs and interconnections of all nodes up to distance k from v. $\mathcal{T}_{v,k}$ is defined to be the topology seen by v after these k rounds, i.e. $\mathcal{T}_{v,k}$ is the graph induced by the k-neighborhood of v where edges between nodes at exactly distance k are excluded. The labelling (i.e. the assignment of IDs) of $\mathcal{T}_{v,k}$ is denoted by $\mathcal{L}(\mathcal{T}_{v,k})$.

Because message size is unbounded, the best a deterministic algorithm can do in time k, is to collect its k-neighborhood and base its decision on $(\mathcal{T}_{v,k}, \mathcal{L}(\mathcal{T}_{v,k}))$. In other words, a deterministic distributed algorithm can be regarded as a function mapping $(\mathcal{T}_{v,k}, \mathcal{L}(\mathcal{T}_{v,k}))$ to the possible outputs. For randomized algorithms, the outcome of v is also dependent on the randomness computed by the nodes in $\mathcal{T}_{v,k}$.

The model presented is in accordance with the one used in [15] and textbooks [19]. It is the strongest possible model when proving lower bounds for local computations because it focuses entirely on the locality of distributed problems and abstracts away other issues arising in the design of distributed algorithms (e.g. need for small messages, fast local computations, etc.). This guarantees that our lower bounds are true consequences of locality.

3. LOWER BOUND

We first give an outline of the proof. The basic idea is to construct a graph $G_k = (V, E)$, for each positive integer k, which contains a bipartite subgraph S with node set $C_0 \cup C_1$ and edges in $C_0 \times C_1$ as shown in Figure 1. Set C_0 consists of n_0 nodes each of which has δ_0 neighbors in C_1. Each of the $n_0 \cdot \frac{\delta_0}{\delta_1}$ nodes in C_1 has δ_1, $\delta_1 > \delta_0$, neighbors in C_0. The goal is to construct G_k in such a way that all nodes in $v \in S$ see the same topology $\mathcal{T}_{v,k}$ within distance k. In a globally optimal solution, all edges of S may be covered by nodes in C_1 and hence, no node in C_0 needs to join the vertex cover.

[1]Our results hold for any possible ID space including the standard case where IDs are the numbers $1, \ldots, n$.

In a local algorithm, however, the decision of whether or not a node joins the vertex cover depends only on its local view, that is, the pair $(\mathcal{T}_{v,k}, \mathcal{L}(\mathcal{T}_{v,k}))$. We show that because adjacent nodes in S see the same $\mathcal{T}_{v,k}$, every algorithm adds a large portion of nodes in C_0 to its vertex cover in order to end up with a feasible solution. In other words, we construct a graph in which the symmetry between two adjacent nodes cannot be broken within k communication rounds. This yields suboptimal local decisions and hence, a suboptimal approximation ratio. Throughout the proof, we will use C_0 and C_1 to denote the two sets of the bipartite subgraph S.

Our proof is organized as follows. The structure of G_k is defined in Subsection 3.1. In Subsection 3.2, we show how G_k can be constructed without small cycles, ensuring that each node sees a tree within distance k. Subsection 3.3 proves that adjacent nodes in C_0 and C_1 have the same view $\mathcal{T}_{v,k}$ and finally, Subsection 3.4 derives the lower bounds.

3.1 The Cluster Tree

The nodes of graph $G_k = (V, E)$ can be grouped into disjoint sets which are linked to each other as bipartite graphs. We call these disjoint sets of nodes *clusters*.

We define the structure of G_k using a directed tree $CT_k = (\mathcal{C}, \mathcal{A})$ with doubly labelled arcs $\ell : \mathcal{A} \to \mathbb{N} \times \mathbb{N}$. We refer to CT_k as the *cluster tree*, because each vertex $C \in \mathcal{C}$ represents a cluster of nodes in G_k. The *size* of a cluster $|C|$ is the number of nodes the cluster contains. An arc $a = (C, D) \in \mathcal{A}$ with $\ell(a) = (\delta_C, \delta_D)$ denotes that the clusters C and D are linked as a bipartite graph, such that each node $u \in C$ has δ_C neighbors in D and each node $v \in D$ has δ_D neighbors in C. It follows that $|C| \cdot \delta_C = |D| \cdot \delta_D$. We call a cluster *leaf-cluster* if it is adjacent to only one other cluster, and we call it *inner-cluster* otherwise.

Definition 3.1. *The cluster tree CT_k is recursively defined as follows:*

$$\begin{aligned}
CT_1 &:= (\mathcal{C}_1, \mathcal{A}_1), \quad \mathcal{C}_1 := \{C_0, C_1, C_2, C_3\} \\
\mathcal{A}_1 &:= \{(C_0, C_1), (C_0, C_2), (C_1, C_3)\} \\
\ell(C_0, C_1) &:= (\delta_0, \delta_1), \quad \ell(C_0, C_2) := (\delta_1, \delta_2), \\
\ell(C_1, C_3) &:= (\delta_0, \delta_1)
\end{aligned}$$

Given CT_{k-1}, we obtain CT_k in two steps:

- *For each inner-cluster C_i, add a new leaf-cluster C'_i with $\ell(C_i, C'_i) := (\delta_k, \delta_{k+1})$.*
- *For each leaf-cluster C_i of CT_{k-1} with $(C_{i'}, C_i) \in \mathcal{A}$ and $\ell(C_{i'}, C_i) = (\delta_p, \delta_{p+1})$, add $k-1$ new leaf-clusters C'_j with $\ell(C_i, C'_j) := (\delta_j, \delta_{j+1})$ for $j = 0 \ldots k, j \neq p+1$.*

Further, we define $|C_0| = n_0$ for all CT_k.

Figure 1 shows CT_2. The shaded subgraph corresponds to CT_1. The labels of each arc $a \in \mathcal{A}$ are of the form $\ell(a) = (\delta_l, \delta_{l+1})$ for some $l \in \{0, \ldots, k\}$. Further, setting $|C_0| = n_0$ uniquely determines the size of all other clusters. In order to simplify the upcoming study of the cluster tree, we need two additional definitions. The *level* of a cluster is the distance to C_0 in the cluster tree (cf. Figure 1). The *depth* of a cluster C is its distance to the furthest leaf in the subtree rooted at C. Hence, the depth of a cluster plus one equals the height of the subtree corresponding to C. In the example of Figure 1, the depths of C_0, C_1, C_2, and C_3 are 3, 2, 1, and 1, respectively.

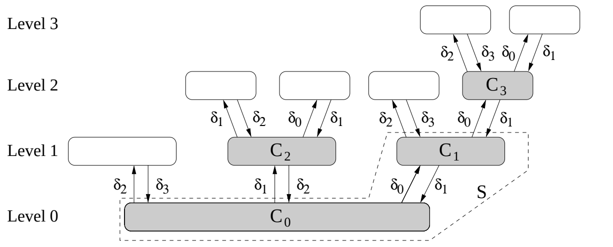


Figure 1: Cluster-Tree CT_2.

Note that CT_k describes the general structure of G_k, i.e. it defines for each node the number of neighbors in each cluster. However, CT_k does not specify the actual adjacencies. In the next subsection, we show that G_k can be constructed so that each node's view is a tree.

3.2 The Lower Bound Graph

In Subsection 3.3, we will prove that the topologies seen by nodes in C_0 and C_1 are identical. This task is greatly simplified if each node's topology is a tree (rather than a general graph) because we do not have to worry about cycles. The *girth* of a graph G, denoted by $g(G)$, is the length of the shortest cycle in G. We want to construct G_k with girth at least $2k+1$ so that in k communication rounds, all nodes see a tree. Given the structural complexity of G_k for large k, constructing G_k with large girth is not a trivial task. The solution we present is based on the construction of the graph family $D(r, q)$ as proposed in [13]. For given r and q, $D(r, q)$ defines a bipartite graph with $2q^r$ nodes and girth $g(D(r, q)) \geq r + 5$. In particular, we show that for appropriate r and q, we obtain an instance of G_k by deleting some of the edges of $D(r, q)$. In the following, we introduce $D(r, q)$ up to the level of detail which is necessary to understand our results. For the interested reader, we refer to [13].

For an integer $r \geq 1$ and a prime power q, $D(r, q)$ defines a bipartite graph with node set $P \cup L$ and edges $E_D \subset P \times L$. The nodes of P and L are labelled by the r-vectors over the finite field $\mathbb{F}_q$, i.e. $P = L = \mathbb{F}_q^r$. In accordance with [13], we denote a vector $p \in P$ by (p) and a vector $l \in L$ by $[l]$. The components of (p) and $[l]$ are written as follows (for $D(r, q)$, the vectors are projected onto the first r coordinates):

$$\begin{aligned}
(p) &= (p_1, p_{1,1}, p_{1,2}, p_{2,1}, p_{2,2}, p'_{2,2}, p_{2,3}, p_{3,2}, \ldots \\
&\quad\; p_{i,i}, p'_{i,i}, p_{i,i+1}, p_{i+1,i}, \ldots) \qquad (1) \\
[l] &= [l_1, l_{1,1}, l_{1,2}, l_{2,1}, l_{2,2}, l'_{2,2}, l_{2,3}, l_{3,2}, \ldots \\
&\quad\; l_{i,i}, l'_{i,i}, l_{i,i+1}, l_{i+1,i}, \ldots]. \qquad (2)
\end{aligned}$$

Note that the somewhat confusing numbering of the components of (p) and $[l]$ is chosen in order to simplify the following system of equations. There is an edge between two nodes (p) and $[l]$, exactly if the first $r-1$ of the following conditions hold (for $i = 2, 3, \ldots$).

$$\begin{aligned}
l_{1,1} - p_{1,1} &= l_1 p_1 \\
l_{1,2} - p_{1,2} &= l_{1,1} p_1 \\
l_{2,1} - p_{2,1} &= l_1 p_{1,1} \\
l_{i,i} - p_{i,i} &= l_1 p_{i-1,i} \qquad (3) \\
l'_{i,i} - p'_{i,i} &= l_{i,i-1} p_1 \\
l_{i,i+1} - p_{i,i+1} &= l_{i,i} p_1 \\
l_{i+1,i} - p_{i+1,i} &= l_1 p'_{i,i}
\end{aligned}$$

In [13], it is shown that for odd $r \geq 3$, $D(r,q)$ has girth at least $r+5$. Further, if a node u and a coordinate of a neighbor v is fixed, the remaining coordinates of v are uniquely determined. This is concretized in the next lemma.

LEMMA 3.2. *For all $(p) \in P$ and $l_1 \in \mathbb{F}_q$, there is exactly one $[l] \in L$ such that l_1 is the first coordinate of $[l]$ and such that (p) and $[l]$ are connected by an edge in $D(r,q)$. Analogously, if $[l] \in L$ and $p_1 \in \mathbb{F}_q$ are fixed, the neighbor (p) of $[l]$ is uniquely determined.*

PROOF. The first $r-1$ equations of (3) define a linear system for the unknown coordinates of $[l]$. If the equations and variables are written in the given order, the matrix corresponding to the resulting linear system of equations is a lower triangular matrix with non-zero elements in the diagonal. Hence, the matrix has full rank and by the basic laws of (finite) fields, the solution is unique. Exactly the same argumentation holds for the second claim of the lemma. $\square$

We are now ready to construct G_k with large girth. We start with an arbitrary instance G'_k of the cluster tree which may have the minimum possible girth 4. An elaboration of the construction of G'_k is deferred to Subsection 3.4. For now, we simply assume that G'_k exists. Both G_k and G'_k are bipartite graphs with odd-level clusters in one set and even-level clusters in the other. Let m be the number of nodes in the larger of the two partitions of G'_k. We choose q to be the smallest prime power greater than or equal to m. In both partitions $V_1(G'_k)$ and $V_2(G'_k)$ of G'_k, we uniquely label all nodes v with elements $c(v) \in \mathbb{F}_q$.

As already mentioned, G_k is constructed as a subgraph of $D(r,q)$ for appropriate r and q. We choose q as described above and we set $r = 2k-4$ such that $g(D(r,q)) \geq 2k+1$. Let $(p) = (p_1, \dots)$ and $[l] = [l_1, \dots]$ be two nodes of $D(r,q)$. (p) and $[l]$ are connected by an edge in G_k if and only if they are connected in $D(r,q)$ and there is an edge between nodes $u \in V_1(G'_k)$ and $v \in V_2(G'_k)$ for which $c(u) = p_1$ and $c(v) = l_1$. Finally, nodes without incident edges are removed from G_k.

LEMMA 3.3. *The graph G_k constructed as described above is a cluster tree with the same degrees δ_i as in G'_k. G_k has at most $2mq^{2k-5}$ nodes and girth at least $2k+1$.*

PROOF. The girth directly follows from the construction; removing edges cannot create cycles.

For the degrees between clusters, consider two neighboring clusters $C'_i \subset V_1(G'_k)$ and $C'_j \subset V_2(G'_k)$ in G'_k. In G_k, each node is replaced by q^{2k-5} new nodes. The clusters C_i and C_j consist of all nodes (p) and $[l]$ which have their first coordinates equal to the labels of the nodes in C'_i and C'_j, respectively. Let each node in C'_i have δ_α neighbors in C'_j, and let each node in C'_j have δ_β neighbors in C'_i. By Lemma 3.2, nodes in C_i have δ_α neighbors in C_j and nodes in C_j have δ_β neighbors in C_i, too. $\square$

Remark. In [14], it has been shown that $D(r,q)$ is disconnected and consists of at least $q^{\lfloor \frac{r+2}{4} \rfloor}$ isomorphic components which the authors call $CD(r,q)$. Clearly, those components are valid cluster trees as well and we could use one of them for the analysis. As our asymptotic results remain unaffected by this observation, we continue to use G_k as constructed above.

3.3 Equality of Views

In this subsection, we prove that two adjacent nodes in clusters C_0 and C_1 have the same *view*, i.e. within distance k, they see exactly the same topology $\mathcal{T}_{v,k}$. Consider a node $v \in G_k$. Given that v's view is a tree, we can derive its *view-tree* by recursively following all neighbors of v. The proof is largely based on the observation that corresponding subtrees occur in both node's view-tree.

Let C_i and C_j be adjacent clusters in CT_k connected by $\ell(C_i, C_j) = (\delta_l, \delta_{l+1})$, i.e. each node in C_i has δ_l neighbors in C_j, and each node in C_j has δ_{l+1} neighbors in C_i. When traversing a node's view-tree, we say that we *enter* cluster C_j (resp., C_i) over *link* δ_l (resp., δ_{l+1}) from cluster C_i (resp., C_j). Furthermore, we make the following definitions:

DEFINITION 3.4. *The following nomenclature refers to subtrees in the view-tree of a node in G_k.*

- *M_i is the subtree seen upon entering cluster C_0 over a link δ_i.*
- *$B_{i,d,\lambda}$ is a subtree seen upon entering a cluster $C \in \mathcal{C} \setminus \{C_0\}$ over a link δ_i, where C is on level λ and has depth d.*

DEFINITION 3.5. *When entering subtree $B_{i,d,\lambda}$ from a cluster on level $\lambda - 1$ ($\lambda + 1$), we write $B^{\uparrow}_{i,d,\lambda}$ ($B^{\downarrow}_{i,d,\lambda}$). The predicate $\neg$ in $B^{\neg}_{i,d,\lambda}$ denotes that instead of δ_i, the label of the link into this subtree is $\delta_i - 1$.*

The predicate $\neg$ is necessary when, after entering C_j from C_i, we immediately return to C_i on link δ_i. In the view-tree, the edge used to enter C_j connects the current subtree to its parent. Thus, this edge is not available anymore and there are only $\delta_i - 1$ edges remaining to return to C_i. The predicates $\uparrow$ and $\downarrow$ describe from which "direction" a cluster has been entered. As the view-trees of nodes in C_0 and C_1 have to be absolutely identical for our proof to work, we must not neglect these admittedly tiresome details.

The following example should clarify the various definitions. Additionally, you may refer to the example of G_3 in Figure 3 in the appendix.

EXAMPLE 3.6. *Consider G_1. Let V_{C_0} and V_{C_1} denote the view-trees of nodes in C_0 and C_1, respectively:*

$$
\begin{array}{llll}
V_{C_0} & = B^{\uparrow}_{0,1,1} \cup B^{\uparrow}_{1,0,1} & V_{C_1} & = B^{\uparrow}_{0,0,2} \cup M_1 \\
B^{\uparrow}_{0,1,1} & = B^{\uparrow}_{0,0,2} \cup M^{\neg}_1 & B^{\uparrow}_{0,0,2} & = B^{\downarrow,\neg}_{1,1,1} \\
B^{\uparrow}_{1,0,1} & = M^{\neg}_2 & M_1 & = B^{\uparrow,\neg}_{0,1,1} \cup B^{\uparrow}_{1,0,1} \\
\dots & & \dots &
\end{array}
$$

We start the proof by giving a set of rules which describe the subtrees seen at a given point in the view-tree. We call these rules *derivation rules* because they allow us to *derive* the view-tree of a node by mechanically applying the matching rule for a given subtree.

LEMMA 3.7. *The following derivation rules hold in G_k:*

$$M_i = \bigcup_{\substack{j=0\dots k \\ j \neq i-1}} B^{\uparrow}_{j,k-j,1} \cup B^{\uparrow,\neg}_{i-1,k-i+1,1}$$

$$B^{\uparrow}_{i,d,1} = F_{\{i+1\}} \cup D_{\{\}} \cup M^{\neg}_{i+1}$$

$$B^{\downarrow}_{i,d,1} = F_{\{i-1,k-d+1\}} \cup D_{\{\}} \cup M_{k-d+1} \cup B^{\uparrow,\neg}_{i-1,d-1,2}$$

$$B^{\uparrow}_{i,d,\lambda} = F_{\{i+1\}} \cup D_{\{i+1\}} \cup B^{\downarrow,\neg}_{i+1,d+1,\lambda-1}$$

where F and D are defined as

$$\begin{aligned} F_W &:= \bigcup_{\substack{j=0\ldots k-d+1\\ j\notin W}} B^{\uparrow}_{j,d-1,\lambda+1} \\ D_W &:= \bigcup_{\substack{j=k-d+2\ldots k\\ j\notin W}} B^{\uparrow}_{j,k-j,\lambda+1}. \end{aligned}$$

PROOF. We first show the derivation rule for M_i. It can be seen in Example 3.6 that the rule holds for $k = 1$. For the induction step, we build CT_{k+1} from CT_k as defined in Definition 3.1. $M^{(k)}$ is an inner cluster and therefore, one new cluster $B_{k+1,0,1}$ is added. The depth of all other subtrees increases by 1 and $M^{(k+1)} := \bigcup_{j=0\ldots k+1} B^{\uparrow}_{j,k-j,1}$ follows. If we enter $M^{(k+1)}$ over link δ_i, there will be only $\delta_{i-1} - 1$ edges left to return to the cluster from which we had entered C_0. Consequently, the link δ_{i-1} features the $\neg$ predicate.

The remaining rules follow along the same lines. Let C_i be a cluster with entry-link δ_i which was first created in CT_r, $r < k$ (Note that in CT_k, $r = k - d$ holds because each subtree increases its depth by one in each "round"). According to the second building rule of Definition 3.1, r new neighboring clusters (subtrees) are created in CT_{r+1}. More precisely, a new cluster is created for all entry-links $\delta_0 \ldots \delta_r$, except δ_i. We call these subtrees *fixed-depth* subtrees F. If the subtree with root C_i has depth d in CT_k, the fixed-depth subtrees have depth $d - 1$. In each $CT_{r'}$, $r' \in \{r + 2, \ldots, k\}$, C_i is an inner-cluster and hence, one new neighboring cluster with entry-link $\delta_{r'}$ is created. We call these subtrees *diminishing-depth* subtrees D. In CT_k, each of these subtrees has grown to depth $k - r'$.

We now turn our attention to the differences between the three rules. They stem from the exceptional treatment of level 1, as well as the predicates $\uparrow$ and $\downarrow$. In $B^{\uparrow}_{i,d,1}$, the link δ_{i+1} returns to C_0, but contains only $\delta_{i+1} - 1$ edges in the view-tree. In $B^{\downarrow}_{i,d,1}$, we have to consider two special cases. The first one is the link to C_0. For a cluster on level 1 with entry-link (from C_0) i, the equality $k = d + i$ holds and therefore, the link to C_0 is δ_{k-d+1} and thus, M_{k-d+1} follows. Secondly, we write $B^{\uparrow,\neg}_{i-1,d-1,2}$, because there is one edge less leading back to the cluster where we had come from. (Note that since we entered the current cluster from a higher level, the link leading back to where we came from is δ_{i-1}, instead of δ_{i+1}). Finally in $B^{\uparrow}_{i,d,\lambda}$, we again have to treat the returning link δ_{i+1} specially.

Note that the general derivation rule for $B^{\downarrow}_{i,d,\lambda}$ is missing as we will not need it for the proof. □

Next, we define the notion of *r-equality.* Intuitively, if two view-trees are r-equal, they have the same topology within distance r.

DEFINITION 3.8. *Let $V_1 = \bigcup_{i=0\ldots k} b_i$ and $V_2 = \bigcup_{i=0\ldots k} b'_i$ be view-trees; b_i and b'_i are subtrees entered on link δ_i. Then, V_1 and V_2 are r-equal if all corresponding subtrees are $(r{-}1)$-equal,*

$$V_1 \overset{r}{=} V_2 \Longleftarrow b_i \overset{r-1}{=} b'_i \ , \ \forall i \in \{0, \ldots, k\}.$$

Further, all subtrees are 0-equal: $B_{i,d,\lambda} \overset{0}{=} B_{i',d',\lambda'}$ and $B_{i,d,\lambda} \overset{0}{=} M_{i'}$ for all i, i', d, d', λ, and λ'.

Using the notion of r-equality, it is now easy to define what we actually have to prove. We will show that in G_k, $V_{C_0} \overset{k}{=} V_{C_1}$ holds. This is equivalent to showing that each node in C_0 sees exactly the same topology within distance k as its neighbor in C_1. We will now establish several helper lemmas.

LEMMA 3.9. *Let β and β' be sets of subtrees, and let $V_{v_1} = B^{\uparrow}_{i,d,x} \cup \beta$ and $V_{v_2} = B^{\uparrow}_{i,d,y} \cup \beta'$ be two view-trees. Then, for all x and y*

$$V_{v_1} \overset{r}{=} V_{v_2} \Longleftarrow \beta \overset{r-1}{=} \beta'.$$

PROOF. Assume that the roots of the subtree of V_{v_1} and V_{v_2} are C_i and C_j. The subtrees have equal depth and entry-link and they have thus grown identically. Hence, all paths which do not return to clusters C_i and C_j must be identical. Further, consider all paths which, after s hops, return to C_i and C_j over link δ_{i+1}. After these s hops, they return to the original cluster and see views V'_{v_1} and V'_{v_2}, differing from V_{v_1} and V_{v_2} only in the placement of the $\neg$ predicate. This does not affect β and β' and therefore,

$$V_{v_1} \overset{r}{=} V_{v_2} \Longleftarrow V'_{v_1} \overset{r-s}{=} V'_{v_2} \ \wedge \ \beta \overset{r-1}{=} \beta' \ , \ s > 1.$$

The same argument can be repeated until $r - s = 0$ and because $V'_{v_1} \overset{0}{=} V'_{v_2}$, the lemma follows. □

LEMMA 3.10. *Let β and β' be sets of subtrees, and let $V_{v_1} = B^{\uparrow}_{i,d,x} \cup \beta$ and $V_{v_2} = B^{\uparrow}_{i,d',y} \cup \beta'$ be two view-trees. Then, for all x and y, and for all $r \leq \min(d, d')$,*

$$V_{v_1} \overset{r}{=} V_{v_2} \Longleftarrow \beta \overset{r-1}{=} \beta'.$$

PROOF. W.l.o.g, we assume $d' \leq d$. In the construction process of G_k, the root clusters of V_{v_1} and V_{v_2} have been created in steps $k - d$ and $k - d'$, respectively. By Definition 3.1, all subtrees with depth $d^* < d'$ have grown identically in both views. The remaining subtrees of V_{v_2} were all created in step $k - d' + 1$ and have depth $d' - 1$. The corresponding subtrees in V_{v_1} have at least the same depth and hence, each pair of corresponding subtrees are $(d' - 1)$-equal. It follows that for $r \leq \min(d, d')$, the subtrees $B^{\uparrow}_{i,d,x}$ and $B^{\uparrow}_{i,d',y}$ are identical within distance r. Using the same argument as in Lemma 3.9 concludes the proof. □

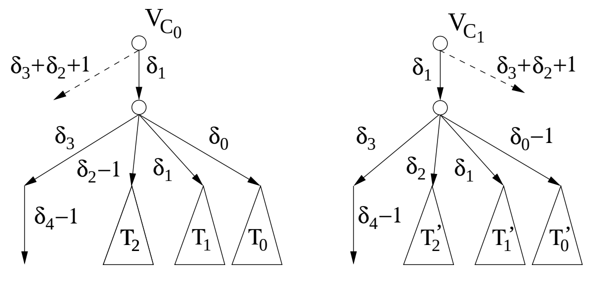


Figure 2: The view-trees V_{C_0} and V_{C_1} in G_3 seen upon using link δ_1.

Figure 2 shows a part of the view-trees of nodes in C_0 and C_1 in G_3. The figure shows that the subtrees with links δ_0 and δ_2 cannot be matched directly to one another because of the different placement of the -1. It turns out that this inherent difference appears in every step of our theorem. However, the following lemma shows that the subtrees T_0 and T_2 (T'_0 and T'_2) are equal up to the required distance and hence, nodes are unable to distinguish them. It is this crucial property of our cluster tree, which allows us to "move" the $\neg$ predicate between links δ_i and δ_{i+2} and enables us to derive the main theorem.

LEMMA 3.11. *Let β and β' be sets of subtrees and let V_{v_1} and V_{v_2} be defined as*

$$\begin{aligned} V_{v_1} &= M_i^{\neg} \cup B^{\uparrow}_{i-2,k-i,2} \cup \beta \\ V_{v_2} &= M_i \cup B^{\uparrow,\neg}_{i-2,k-i,2} \cup \beta'. \end{aligned}$$

Then, for all $i \in \{2,\dots,k\}$,

$$V_{v_1} \overset{k-i}{=} V_{v_2} \Longleftarrow \beta \overset{k-i-1}{=} \beta'.$$

PROOF. Again, we make use of Lemma 3.7 to show that M_i and $B^{\uparrow}_{i-2,k-i,2}$ are (k–i–1)-equal. The claim then follows from the fact that the two subtrees are not distinguishable and the placement of the $\neg$ predicate is irrelevant.

$$\begin{aligned} M_i &= \bigcup_{\substack{j=0\dots k\\ j\neq i-1}} B^{\uparrow}_{j,k-j,1} \cup B^{\uparrow,\neg}_{i-1,k-i+1,1} \\ B^{\uparrow}_{i-2,k-i,2} &= \bigcup_{\substack{j=0\dots i+1\\ j\neq i-1}} B^{\uparrow}_{j,k-i-1,3} \cup \bigcup_{j=i+2\dots k} B^{\uparrow}_{j,k-j,3} \\ &\quad \cup \ B^{\downarrow,\neg}_{i-1,k-i+1,1} \end{aligned}$$

For $j = \{0,\dots,i-2,i,\dots,k\}$, all subtrees are equal according to Lemmas 3.9 and 3.10. It remains to be shown that $B^{\uparrow}_{i-1,k-i+1,1} \overset{k-i-2}{=} B^{\downarrow}_{i-1,k-i+1,1}$. For that purpose, we plug $B^{\uparrow}_{i-1,k-i+1,1}$ and $B^{\downarrow}_{i-1,k-i+1,1}$ into Lemma 3.7 and show their equality using the derivation rules. Let β be defined as $\beta := F_{\{i-2,i\}} \cup D_{\{\}}$.

$$\begin{aligned} B^{\uparrow}_{i-1,k-i+1,1} &= F_{\{i\}} \cup D_{\{\}} \cup M_i^{\neg} \\ &= B^{\uparrow}_{i-2,k-i,2} \cup M_i^{\neg} \cup \beta \\ B^{\downarrow}_{i-1,k-i+1,1} &= F_{\{i-2,i\}} \cup D_{\{\}} \cup M_i \cup B^{\uparrow,\neg}_{i-2,k-i,2} \\ &= B^{\uparrow,\neg}_{i-2,k-i,2} \cup M_i \cup \beta \end{aligned}$$

Again, if M_i and $B^{\uparrow}_{i-2,k-i,2}$ are (k–i–3)-equal, we can move the $\neg$ predicate because the subtrees are indistinguishable. Hence, we have to show $M_i \overset{k-i-3}{=} B^{\uparrow}_{i-2,k-i,2}$. In the proof, we have reduced $V_{v_1} \overset{k-i}{=} V_{v_2}$ stepwise to an expression of diminishing equality conditions, i.e.

$$\begin{aligned} V_{v_1} \overset{k-i}{=} V_{v_2} &\Longleftarrow M_i \overset{k-i-1}{=} B^{\uparrow}_{i-2,k-i,2} \\ &\Longleftarrow B^{\uparrow}_{i-1,k-i+1,1} \overset{k-i-2}{=} B^{\downarrow}_{i-1,k-i+1,1} \\ &\Longleftarrow M_i \overset{k-i-3}{=} B^{\uparrow}_{i-2,k-i,2}. \end{aligned}$$

This process can be continued until either

$$B^{\uparrow}_{i-1,k-i+1,1} \overset{0}{=} B^{\downarrow}_{i-1,k-i+1,1} \quad \text{or} \quad M_i \overset{0}{=} B^{\uparrow}_{i-2,k-i,2}$$

which is always true. □

Finally, we are ready to prove the main theorem.

THEOREM 3.12. *Consider graph G_k. Let V_{C_0} and V_{C_1} be the view-trees of two adjacent nodes in clusters C_0 and C_1, respectively. Then, $V_{C_0} \overset{k}{=} V_{C_1}$.*

PROOF. Initially, each node in C_0 sees subtree M_* and each node in C_1 sees $B_{*,k,1}$ ($*$ denotes that the subtree has not been entered on any link):

$$\begin{aligned} V_{C_0} \quad &: \quad M_* = \bigcup_{j=0\dots k} B^{\uparrow}_{j,k-j,1} \\ V_{C_1} \quad &: \quad B_{*,k,1} = \bigcup_{\substack{j=0\dots k\\ j\neq 1}} B^{\uparrow}_{j,k-j,2} \cup M_1. \end{aligned}$$

It follows $V_{C_0} \overset{k}{=} V_{C_1} \Longleftarrow B^{\uparrow}_{1,k-1,1} \overset{k-1}{=} M_1$ because all other subtrees are $(k-1)$-equal by Lemma 3.9. Having reduced $V_{C_0} \overset{k}{=} V_{C_1}$ to $B^{\uparrow}_{1,k-1,1} \overset{k-1}{=} M_1$, we can further reduce it to $M_2 \overset{k-2}{=} B^{\uparrow}_{2,k-2,1}$:

$$\begin{aligned} M_1 \quad &= \quad \bigcup_{j=1\dots k} B^{\uparrow}_{j,k-j,1} \cup B^{\uparrow,\neg}_{0,k,1} \\ B^{\uparrow}_{1,k-1,1} \quad &= \quad B^{\uparrow}_{0,k-2,2} \cup B^{\uparrow}_{1,k-2,2} \cup D_{\{\}} \cup M_2^{\neg} \\ &\underset{\text{Lem. 3.11}}{\overset{k-2}{=}} \quad B^{\uparrow,\neg}_{0,k-2,2} \cup B^{\uparrow}_{1,k-2,2} \cup D_{\{\}} \cup M_2. \end{aligned}$$

By Lemmas 3.9 and 3.10, all subtree are $(k-2)$-equal, except $B^{\uparrow}_{2,k-2,1}$ and M_2.

It seems clear that we can continue to reduce $V_{C_0} \overset{k}{=} V_{C_1}$ step by step in the same fashion until we reach 0. For the induction step, we assume $V_{C_0} \overset{k}{=} V_{C_1} \Longleftarrow B^{\uparrow}_{r,k-r,1} \overset{k-r}{=} M_r$ for $r < k$ and prove $V_{C_0} \overset{k}{=} V_{C_1} \Longleftarrow B^{\uparrow}_{r+1,k-r-1,1} \overset{k-r-1}{=} M_{r+1}$.

$$\begin{aligned} M_r \quad &= \quad \bigcup_{\substack{j=0\dots k\\ j\neq r-1}} B^{\uparrow}_{j,k-j,1} \cup B^{\uparrow,\neg}_{r-1,k-r+1,1} \\ B^{\uparrow}_{r,k-r,1} \quad &= \quad \bigcup_{j=0\dots r} B^{\uparrow}_{j,k-r-1,2} \cup D_{\{\}} \cup M_{r+1}^{\neg} \\ &\underset{\text{Lem. 3.11}}{\overset{k-r-1}{=}} \quad \bigcup_{\substack{j=0\dots r\\ j\neq r-1}} B^{\uparrow}_{j,k-r-1,2} \cup B^{\uparrow,\neg}_{r-1,k-r-1,2} \\ &\qquad \cup \bigcup_{j=r+2\dots k} B^{\uparrow}_{j,k-j,2} \cup M_{r+1}. \end{aligned}$$

Apart from M_{r+1} (resp,. $B^{\uparrow}_{r+1,k-r-1,1}$), all subtrees are $(k-r-1)$-equal by Lemmas 3.9 and 3.10. Since M_{r+1} and $B^{\uparrow}_{r+1,k-r-1,1}$ are the only subtrees not being immediately matched, the induction step follows. For $r = k-1$, we get $V_{C_0} \overset{k}{=} V_{C_1} \Longleftarrow B^{\uparrow}_{k,0,1} \overset{0}{=} M_k$, which concludes the proof because $B^{\uparrow}_{k,0,1} \overset{0}{=} M_k$ is true. □

Remark. As a side-effect, the proof of Theorem 3.12 has highlighted the fundamental significance of the *critical path* $P = (\delta_1, \delta_2, \dots, \delta_k)$ in CT_k. After following path P, the view of a node $v \in C_0$ ends up in the leaf-cluster neighboring C_0 and sees δ_{i+1} neighbors. Following the same path, a node $v' \in C_1$ ends up in C_0 and sees $\sum_{j=0}^{i} \delta_j - 1$ neighbors. There is no way to match these views. This inherent inequality is the underlying reason for the way G_k is defined: It must be ensured that the critical path is at least k hops long.

3.4 Analysis

In this subsection, we derive the lower bounds on the approximation ratio of k-local MVC algorithms. Let OPT be an optimal solution for MVC and let ALG be the solution computed by any algorithm. The main observation is that adjacent nodes in the clusters C_0 and C_1 have the same view and therefore, every algorithm treats nodes in both of the two clusters the same way. Consequently, ALG contains a significant portion of the nodes of C_0, whereas the optimal solution covers the edges between C_0 and C_1 entirely by nodes in C_1.

LEMMA 3.13. *Let ALG be the solution of any distributed (randomized) vertex cover algorithm which runs for at most k rounds. When applied to G_k as constructed in Subsection 3.2 in the worst case (in expectation), ALG contains at least half of the nodes of C_0.*

PROOF. Let $v_0 \in C_0$ and $v_1 \in C_1$ be two arbitrary, adjacent nodes from C_0 and C_1. We first prove the lemma for deterministic algorithms. The decision whether a given node v enters the vertex cover depends solely on the topology $\mathcal{T}_{v,k}$ and the labelling $\mathcal{L}(\mathcal{T}_{v,k})$. Assume that the labelling of the graph is chosen uniformly at random. Further, let $p_0^{\mathcal{A}}$ and $p_1^{\mathcal{A}}$ denote the probabilities that v_0 and v_1, respectively, end up in the vertex cover when a deterministic algorithm $\mathcal{A}$ operates on the randomly chosen labelling. By Theorem 3.12, v_0 and v_1 see the same topologies, that is, $\mathcal{T}_{v_0,k} = \mathcal{T}_{v_1,k}$. With our choice of labels, v_0 and v_1 also see the same distribution on the labellings $\mathcal{L}(\mathcal{T}_{v_0,k})$ and $\mathcal{L}(\mathcal{T}_{v_1,k})$. Therefore it follows that $p_0^{\mathcal{A}} = p_1^{\mathcal{A}}$.

We have chosen v_0 and v_1 such that they are neighbors in G_k. In order to obtain a feasible vertex cover, at least one of the two nodes has to be in it. This implies $p_0^{\mathcal{A}} + p_1^{\mathcal{A}} \geq 1$ and therefore $p_0^{\mathcal{A}} = p_1^{\mathcal{A}} \geq 1/2$. In other words, for all nodes in C_0, the probability to end up in the vertex cover is at least 1/2. Thus, by the linearity of expectation, at least half of the nodes of C_0 are chosen by algorithm $\mathcal{A}$. Therefore, for every deterministic algorithm $\mathcal{A}$, there is at least one labelling for which at least half of the nodes of C_0 are in the vertex cover.[2]

The argument for randomized algorithms is now straightforward using Yao's minimax principle. The expected number of nodes chosen by a randomized algorithm cannot be smaller than the expected number of nodes chosen by an optimal deterministic algorithm for an arbitrarily chosen distribution on the labels.

□

Lemma 3.13 gives a lower bound on the number of nodes chosen by any k-local MVC algorithm. In particular, we have that $\mathrm{E}[|ALG|] \geq |C_0|/2 = n_0/2$. We do not know OPT, but since the nodes of cluster C_0 are not necessary to obtain a feasible vertex cover, the optimal solution is bounded by $|OPT| \leq n - n_0$. In the following, we define

$$\delta_i := \delta^i \quad , \ \forall i \in \{0, \ldots, k+1\} \tag{4}$$

for some value δ.

LEMMA 3.14. *If $k+1 < \delta$, the number of nodes n of G_k is*

$$n \ \leq \ n_0 \left(1 + \frac{k+1}{\delta - (k+1)}\right).$$

PROOF. There are n_0 nodes in C_0. By (4), the number of nodes per cluster decreases for each additional level by a factor δ. Hence, a cluster on level l contains n_0/δ^l nodes. By the definition of CT_k, each cluster has at most $k+1$ neighboring clusters on a higher level. Thus, the number of nodes n_l on level l is upper bounded by

$$n_l \ \leq \ (k+1)^l \cdot \frac{n_0}{\delta^l}.$$

Summing up over all levels l and interpreting the sum as a geometric series, we obtain

$$\begin{aligned} n \ &\leq \ n_0 \cdot \sum_{i=0}^{k+1} \left(\frac{k+1}{\delta}\right)^l \ \leq \ n_0 \cdot \sum_{i=0}^{\infty} \left(\frac{k+1}{\delta}\right)^l \\ &= \ n_0 + n_0 \left(\frac{k+1}{\delta}\right)\left(\frac{1}{1 - \frac{k+1}{\delta}}\right) \\ &= \ n_0 \left(1 + \frac{k+1}{\delta - (k+1)}\right). \end{aligned}$$

□

It remains to determine the relationship between δ and n_0 such that G_k can be realized as described in Subsection 3.2. There, the construction of G_k with large girth is based on a smaller instance G'_k where girth does not matter. Using (4) (i.e. $\delta_i := \delta^i$), we can now tie up this loose end and describe how to obtain G'_k. The number of nodes per cluster decreases by a factor δ on each level of CT_k. Including C_0, CT_k consists of $k+2$ levels. The maximum number of neighbors inside a leaf-cluster is δ^k. Hence, we can set the sizes of the clusters on the outermost level $k+1$ to be δ^k. This implies that the size of a cluster on level l is δ^{2k+1-l}. Particularly, the size of C'_0 at level 0 in G'_k is $n'_0 = \delta^{2k+1}$. Let C_i and C_j be two adjacent clusters with $\ell(C_i, C_j) = (\delta^i, \delta^{i+1})$. C_i and C_j can simply be connected by as many complete bipartite graphs $K_{\delta^i,\delta^{i+1}}$ as necessary.

If we assume that $k+1 \leq \delta/2$, we have $n \leq 2n_0$ by Lemma 3.14. Applying the construction of Subsection 3.2, we get $n_0 \leq n'_0 \cdot \langle n' \rangle^{2k-5}$, where $\langle n' \rangle$ denotes the smallest prime power larger than or equal to n', i.e. $\langle n' \rangle < 4n'_0$. Putting all together, we get

$$n_0 \ \leq \ (4n'_0)^{2k-4} \ \leq \ 4^{2k-4}\delta^{4k^2}. \tag{5}$$

THEOREM 3.15. *There are graphs G, such that in k communication rounds, every distributed algorithm for the minimum vertex cover problem on G has approximation ratios at least*

$$\Omega\left(\frac{n^{c/k^2}}{k}\right) \quad \textit{and} \quad \Omega\left(\frac{\Delta^{1/k}}{k}\right)$$

for some constant $c \geq 1/4$, where n and Δ denote the number of nodes and the highest degree in G, respectively.

PROOF. We can choose $\delta \geq 4^{-1/(2k)} n_0^{1/(4k^2)}$ due to Inequality (5). Finally, using Lemmas 3.13 and 3.14, the approximation ratio α is at least

$$\begin{aligned} \alpha \ &\geq \ \frac{n_0/2}{n - n_0} \ \geq \ \frac{n_0/2 \cdot \delta/2}{n_0 \cdot (k+1)} \ = \ \frac{\delta}{4(k+1)} \\ &\geq \ \frac{(n/2)^{1/(4k^2)}}{4^{1+1/(2k)}(k+1)} \ \in \ \Omega\left(\frac{n^{1/(4k^2)}}{k}\right). \end{aligned}$$

The second lower bound follows from $\Delta = \delta^{k+1}$. □

[2]In fact, since at most $|C_0|$ such nodes can be in the vertex cover, for at least 1/3 of the labellings, the number exceeds $|C_0|/2$.

THEOREM 3.16. *In order to obtain a polylogarithmic or even constant approximation ratio, every distributed algorithm for the MVC problem requires at least* $\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right)$ *and* $\Omega\left(\frac{\log\Delta}{\log\log\Delta}\right)$ *communication rounds.*

PROOF. We set $k = \beta\sqrt{\log n/\log\log n}$ for an arbitrary constant $\beta > 0$. When plugging this into the first lower bound of Theorem 3.15, we get the following approximation ratio α:

$$\alpha \;\geq\; \gamma n^{\frac{c\log\log n}{\beta^2\log n}} \cdot \frac{1}{\beta}\sqrt{\frac{\log\log n}{\log n}}$$

where γ is the hidden constant in the Ω-notation. For the logarithm of α, we get

$$\begin{aligned}\log\alpha &\geq \frac{c\log\log n}{\beta^2\log n}\cdot\log n - \frac{1}{2}\cdot\log\log n - \log\beta\\ &= \left(\frac{c}{\beta^2}-\frac{1}{2}\right)\cdot\log\log n - \log\beta.\end{aligned}$$

and therefore

$$\alpha \;\in\; \Omega\left(\log(n)^{\left(\frac{c}{\beta^2}-\frac{1}{2}\right)}\right).$$

By choosing an appropriate β, we can determine the exponent of the above expression. For every polylogarithmic term $\alpha(n)$, there is a constant β such that the above expression is at least $\alpha(n)$ and hence, the first lower bound of the theorem follows.

The second lower bound follows from an analogous computation by setting $k = \beta\log\Delta/\log\log\Delta$. □

Remark. By defining $\delta_i := \delta^i$, $i \in \{0,\ldots,k\}$ and $\delta_{k+1} := \delta^{k+1/2}$ (instead of δ^{k+1}), we obtain slightly stronger approximation lower bounds of

$$\Omega\left(n^{c/k^2}-k\right) \quad\text{and}\quad \Omega\left(\Delta^{c'/k}-k\right). \tag{6}$$

The bounds of (6) clearly do not suffice to improve the results of Theorem 3.16.

4. REDUCTIONS

Using the lower bound for vertex cover, we can obtain lower bounds for several other classical graph problems. In this section, we give time lower bounds for the construction of maximal matchings and maximal independent sets as well as for the approximation of minimum dominating set.

A maximal matching (MM) of a graph G is a maximal set of edges which do not share common end-points. Hence, a MM is a set of non-adjacent edges of G such that all edges in $E(G)\setminus MM$ have a common end-point with an edge in MM. A maximal independent set (MIS) is a maximal set of non-adjacent nodes, i.e. all nodes not in the MIS are adjacent to some node of the MIS. The best known lower bound for the distributed computation of a MM or a MIS is $\Omega(\log^* n)$ which holds for rings [15]. Based on Theorem 3.16, we get the following stronger lower bounds.

THEOREM 4.1. *There are graphs G on which every distributed, possibly randomized algorithm requires time*

$$\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right) \quad \textit{and} \quad \Omega\left(\frac{\log\Delta}{\log\log\Delta}\right)$$

to compute a maximal matching. The same lower bounds hold for the construction of maximal independent sets.

PROOF. It is well known that the set of all end-points of the edges of a MM form a 2-approximation for MVC. This simple 2-approximation algorithm is commonly attributed to Gavril and Yannakakis. The lower bound for the construction of a MM therefore directly follows from Theorem 3.16.

For the MIS problem, consider the line graph $L(G_k)$ of G_k. The nodes of a line graph $L(G)$ of G are the edges of G. Two nodes in $L(G)$ are connected by an edge whenever the two corresponding edges in G are incident to the same node. The MM problem on a graph G is equivalent to the MIS problem on $L(G)$. Further, if the real network graph is G, k communication rounds on $L(G)$ can be simulated in $k + \mathrm{O}(1)$ communication rounds on G. Therefore, the times t to compute a MIS on $L(G_k)$ and t' to compute a MM on G_k can only differ by a constant, $t \geq t' - \mathrm{O}(1)$. Let n' and Δ' denote the number of nodes and the maximum degree of G_k, respectively. The number of nodes n of $L(G_k)$ is less than $n'^2/2$, the maximum degree Δ of G_k is less than $2\Delta'$. Because n' only appears as $\log n'$, the power of 2 does not hurt and the theorem holds ($\log n = \Theta(\log n')$). □

We conclude this section by considering the problem of approximating the minimum dominating set (MDS) problem. A dominating set S is a subset of the nodes of a graph G such that all nodes of G are either in S or they have a neighbor in S. In a non-distributed setting, MDS in equivalent to the general minimum set cover problem[3] whereas MVC is a special case of set cover which can be approximated much better. It is therefore not surprising that in a distributed environment, MDS is strictly harder than MVC, too. In the following, we show that this intuitive fact can be formalized.

THEOREM 4.2. *There are graphs G, such that in k communication rounds, every distributed algorithm for the minimum dominating set problem on G has approximation ratios at least*

$$\Omega\left(\frac{n^{c/k^2}}{k}\right) \quad \textit{and} \quad \Omega\left(\frac{\Delta^{1/k}}{k}\right)$$

for some constant c, where n and Δ denote the number of nodes and the highest degree in G, respectively.

PROOF. We show that every MVC instance can be seen as a MDS instance with the same locality. Let $G' = (V', E')$ be a graph for which we want to solve MVC. We construct the corresponding dominating set graph $G = (V, E)$ as follows. For every node and for every edge in G', there is a node in G. We call nodes $v_n \in V$ corresponding to nodes $v' \in V'$ *n-nodes*, and nodes $v_e \in V$ corresponding to edges $e' \in E'$ *e-nodes.* Two n-nodes are connected by an edge if and only if they are adjacent in G'. An n-node v_n and an e-node v_e are connected exactly if the corresponding node and edge are incident in G'. There are no edges between two e-nodes. Clearly, the localities of G' and G are the same, i.e. k communication rounds on one of the two graphs can be simulated by $k + \mathrm{O}(1)$ rounds on the other graph. Let C be a feasible vertex cover for G'. We claim that all nodes of

[3]There exist approximation preserving reductions in both directions.

G corresponding to nodes in C form a valid dominating set on G. By definition, all e-nodes are covered. The remaining nodes of G are covered because for a given graph, a valid vertex cover is a valid dominating set as well. Therefore, the optimal dominating set on G is at most as big as the optimal vertex cover on G'. There also exists a transformation in the other direction. Let D be a valid dominating set on G. If D contains an e-node v_e, we can replace v_e by one of its two neighbors. The size of D remains the same and all three nodes covered (dominated) by v_e are still covered. By this, we get a dominating set D' which has the same size as D and which consists only of n-nodes. Because D' dominates all e-nodes, the nodes of G' corresponding to D' form a valid vertex cover. Thus, MDS on G is exactly as hard as MVC on G' and the theorem follows from Theorem 3.15. □

Corollary 4.3. *To obtain a polylogarithmic or constant approximation ratio for minimum dominating set, there are graphs on which every distributed algorithm needs time*

$$\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right) \quad \textit{and} \quad \Omega\left(\frac{\log\Delta}{\log\log\Delta}\right).$$

Proof. The corollary is a direct consequence of Theorem 4.2 and the proof of Theorem 3.16. □

Remark. Note that in the above corollary, we give a time lower bound for constant MDS approximation although it has been shown that MDS cannot be approximated better than $\ln\Delta$ unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$) [5]. Because local computation is for free in our model, however, it is theoretically possible to get a constant-factor approximation for MDS.

5. CONCLUSIONS

As distributed systems grow larger, it is becoming increasingly vital to design algorithms which do not need to maintain full information about the network. Unfortunately, with a few notable exceptions [15], there have been almost no hard results, which would have shed light into the theoretical possibilities and limitations of locality-based approaches. We have shown locality-imposed restrictions on the approximability and computability of a number of distributed problems. Comparing with the respective upper bounds, some of our lower bounds are near tight. We hope and believe that the various lower bounds given in the present paper will help to ameliorate this situation.

6. REFERENCES

[1] Y. Afek, S. Kutten, and M. Yung. The Local Detection Paradigm and its Applications to Self-Stabilization. *Theoretical Computer Science*, 186(1-2):199–229, 1997.

[2] R. Cole and U. Vishkin. Deterministic Coin Tossing with Applications to Optimal Parallel List Ranking. *Information and Control*, 70(1):32–53, 1986.

[3] M. Elkin. A Faster Distributed Protocol for Constructing a Minimum Spanning Tree. In *Proc. of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 359–368, 2004.

[4] M. Elkin. Unconditional Lower Bounds on the Time-Approximation Tradeoffs for the Distributed Minimum Spanning Tree Problem. In *Proc. of the 36th ACM Symposium on Theory of Computing (STOC)*, 2004.

[5] U. Feige. A Threshold of ln n for Approximating Set Cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[6] F. Fich and E. Ruppert. Hundreds of impossibility results for distributed computing. *Distrib. Comput.*, 16(2-3):121–163, 2003.

[7] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of Distributed Consensus With One Faulty Process. *J. ACM*, 32(2):374–382, 1985.

[8] A. Israeli and A. Itai. A Fast and Simple Randomized Parallel Algorithm for Maximal Matching. *Information Processing Letters*, 22:77–80, 1986.

[9] F. Kuhn and R. Wattenhofer. Constant-Time Distributed Dominating Set Approximation. In *Proc. of the 22nd Annual ACM Symp. on Principles of Distributed Computing (PODC)*, pages 25–32, 2003.

[10] F. Kuhn and R. Wattenhofer. Distributed Combinatorial Optimization. Technical Report 426, ETH Zurich, Dept. of Computer Science, 2003.

[11] E. Kushilevitz and Y. Mansour. An $\Omega(D\log(N/D))$ Lower Bound for Broadcast in Radio Networks. *SIAM Journal on Computing*, 27(3):702–712, June 1998.

[12] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[13] F. Lazebnik and V. A. Ustimenko. Explicit Construction of Graphs with an Arbitrary Large Girth and of Large Size. *Discrete Applied Mathematics*, 60(1-3):275–284, 1995.

[14] F. Lazebnik, V. A. Ustimenko, and A. J. Woldar. A New Series of Dense Graphs of High Girth. *Bulletin of the American Mathematical Society (N.S.)*, 32(1):73–79, 1995.

[15] N. Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[16] Z. Lotker, B. Patt-Shamir, and D. Peleg. Distributed MST for Constant Diameter Graphs. In *Proc. of the 20th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 63–71, 2001.

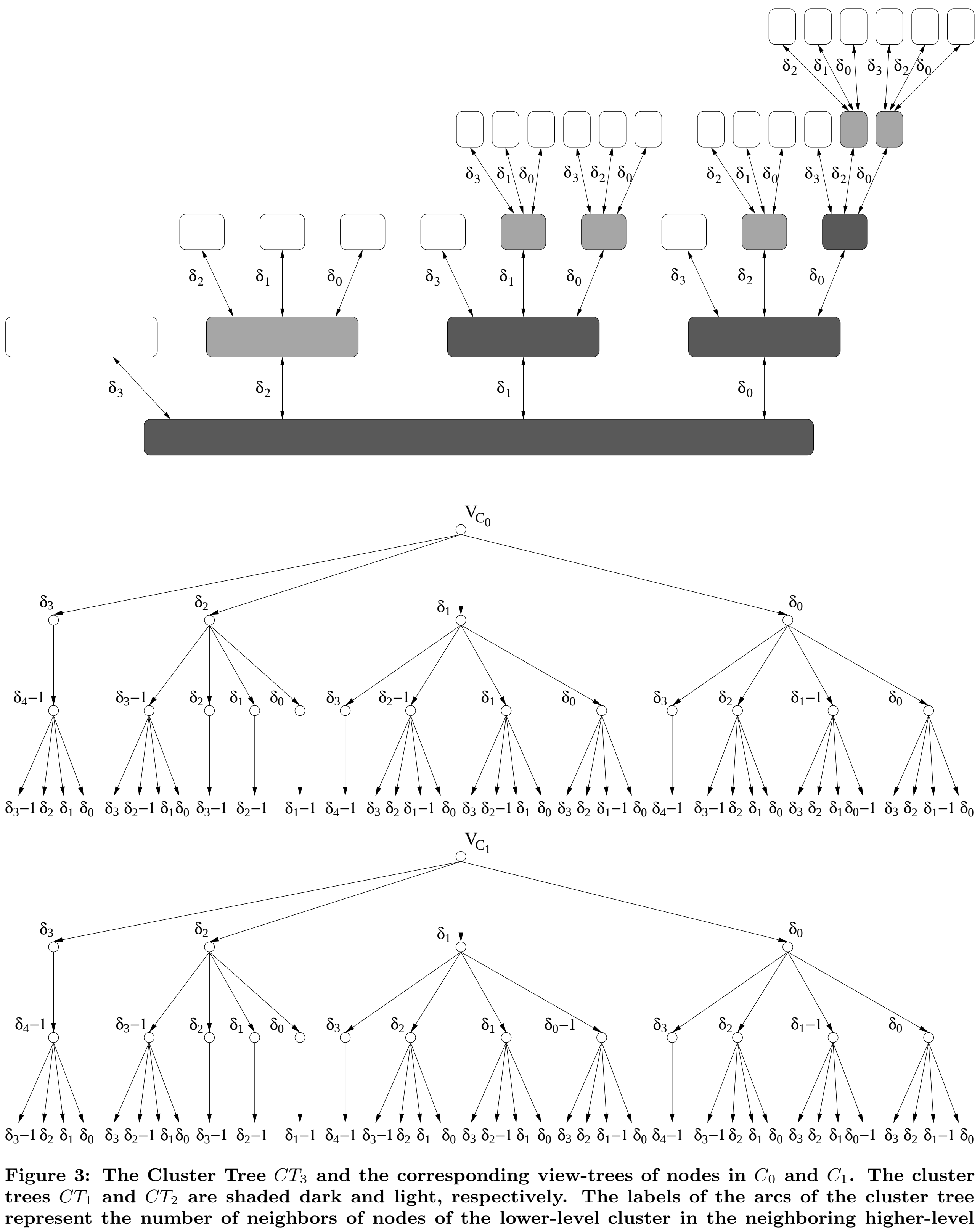
[17] M. Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM Journal on Computing*, 15:1036–1053, 1986.

[18] M. Naor and L. Stockmeyer. What Can Be Computed Locally? In *Proc. of the 25th Annual ACM Symp. on Theory of Computing (STOC)*, pages 184–193, 1993.

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

[20] D. Peleg and V. Rubinovich. A Near-Tight Lower Bound on the Time Complexity of Distributed Minimum-Weight Spanning Tree Construction. *SIAM Journal on Computing*, 30(5):1427–1442, 2000.

APPENDIX



δ2 δ1 δ0 δ3 δ2 δ0
δ3 δ1 δ0 δ3 δ2 δ0 δ2 δ1 δ0 δ3 δ2 δ0
δ2 δ1 δ0 δ3 δ1 δ0 δ3 δ2 δ0
δ3 δ2 δ1 δ0
V_{C_0}
δ3 δ2 δ1 δ0
δ4−1 δ3−1 δ2 δ1 δ0 δ3 δ2−1 δ1 δ0 δ3 δ2 δ1−1 δ0
δ3−1 δ2 δ1 δ0 δ3 δ2−1 δ1 δ0 δ3−1 δ2−1 δ1−1 δ4−1 δ3 δ2 δ1−1 δ0 δ3 δ2−1 δ1 δ0 δ3 δ2 δ1−1 δ0 δ4−1 δ3−1 δ2 δ1 δ0 δ3 δ2 δ1 δ0−1 δ3 δ2 δ1−1 δ0
V_{C_1}
δ3 δ2 δ1 δ0
δ4−1 δ3−1 δ2 δ1 δ0 δ3 δ2 δ1 δ0−1 δ3 δ2 δ1−1 δ0
δ3−1 δ2 δ1 δ0 δ3 δ2−1 δ1 δ0 δ3−1 δ2−1 δ1−1 δ4−1 δ3−1 δ2 δ1 δ0 δ3 δ2−1 δ1 δ0 δ3 δ2 δ1−1 δ0 δ4−1 δ3−1 δ2 δ1 δ0 δ3 δ2 δ1 δ0−1 δ3 δ2 δ1−1 δ0

Figure 3: The Cluster Tree CT_3 and the corresponding view-trees of nodes in C_0 and C_1. The cluster trees CT_1 and CT_2 are shaded dark and light, respectively. The labels of the arcs of the cluster tree represent the number of neighbors of nodes of the lower-level cluster in the neighboring higher-level cluster. The labels of the reverse links are omitted. In the view-trees, an arc labelled with δ_i stands for δ_i edges, all connecting to identical subtrees.