Lower and Upper Bounds for Distributed Packing and Covering

Lower and Upper Bounds for Distributed Packing and Covering

Fabian Kuhn, Thomas Moscibroda, Roger Wattenhofer
{kuhn@inf,moscibroda@student,wattenhofer@inf}.ethz.ch
Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland

Abstract

We make a step towards understanding the distributed complexity of global optimization problems. We give bounds on the trade-off between locality and achievable approximation ratio of distributed algorithms for packing and covering problems. We show that in k communication rounds, maximum matching and therefore packing problems cannot be approximated better than $\Omega(n^{c/k^2}/k)$ and $\Omega(\Delta^{1/k}/k)$ where c is a small constant and n and Δ denote the number of nodes and the maximum degree of the network graph, respectively. This means that $\Omega(\sqrt{\log n/\log\log n})$ and $\Omega(\log\Delta/\log\log\Delta)$ rounds are needed to obtain a constant or polylogarithmic approximation. On the positive side, we prove that maximum matching and minimum vertex cover (the dual problem) can be approximated by $\mathrm{O}(\Delta^{1/k})$ in $\mathrm{O}(k)$ rounds, showing that the given lower bound is almost tight. We also give a distributed algorithm which approximates any packing or covering LP by $\mathrm{O}(n^{1/k})$ in $\mathrm{O}(k)$ rounds.

1 Introduction

Computing a global objective based on local information only lies at the heart of distributed computing theory. In this paper we present the first lower bounds for distributed packing problems such as maximum matching. In addition we exhibit new algorithms for packing (and also covering) problems which almost match the lower bounds.

Throughout the paper we study the standard *message passing* model for distributed computing. Processors of a distributed system are represented by nodes of an undirected graph. Two processors can communicate directly if and only if they are connected by an edge in the graph. We assume that the processors operate in synchrony, that is, communication is round-based, and in each round each node can send a message to all its neighbors in the graph, and receive the messages sent by its neighbors. The *time complexity* of a distributed algorithm is the maximum possible number of rounds needed until every node has completed its computation. In the context of emerging dynamic and mobile distributed systems such as peer-to-peer, ad-hoc, or sensor networks, it is often desirable to keep the time complexity as small as possible even at the cost of somewhat non-optimal global solutions. We are therefore interested in the possible trade-off between time complexity and approximation quality for various (combinatorial) packing and covering problems on the given network graph. In particular, we are interested in algorithms whose time complexity is polylogarithmic or even constant (a.k.a. local algorithms).

In distributed systems the processors need to coordinate themselves; many common coordination tasks boil down to classic graph theory problems such as coloring, dominating or independent set.[1] In this paper we first focus on graph packing problems. In particular, we give lower bounds on the trade-off between time complexity and approximation quality for maximum matching[2] (MM). A matching in a graph is a subset of edges such that no two edges in the matching are adjacent. A maximum matching is a matching of maximum cardinality.

The best known approximation algorithm for distributed maximum matching is due to Israeli and

[1] A coloring allows nodes in wireless networks to establish an efficient frequency division multiplexing scheme. Dominating sets are traditionally used for clustering, independent sets for parallel execution.

[2] Matching for example allows nodes of a wireless network to start non-interfering communication channels.

Itai [6]. Using a simple randomized mechanism, a maximal matching[3] is computed in $O(\log n)$ expected time. Since a maximal matching is a 2-approximation of a maximum matching[4] we have a constant approximation in only a logarithmic number of rounds.

Since no lower bounds are known, it is natural to ask whether one could also achieve a constant approximation in constant time only. Indeed for special graphs such as trees constant-time constant-approximation algorithms are known [19]. Also for regular graphs a distributed randomized rounding technique will give a constant approximation in constant time [10]. Fooling around with the usual suspect graphs in distributed computing (e.g. ring, mesh) indicates that a maximum matching approximation is indeed simpler than a maximal matching, which seems to require logarithmic time because of symmetry breaking.

In this paper we show that in general this intuition might be misleading. In particular we prove that in k communication rounds, maximum matching (and even its fractional version) cannot be approximated within $\Omega(n^{c/k^2}/k)$ and $\Omega(\Delta^{1/k}/k)$ for some constant c, n and Δ denoting the number of nodes and the largest degree in the graph, respectively. These results imply that a minimum of $\Omega(\sqrt{\log n/\log\log n})$ and $\Omega(\log\Delta/\log\log\Delta)$ communication rounds are required in order to obtain a constant or polylogarithmic approximation. Since these lower bounds hold even in the cases of unbounded message-size and complete synchrony, they are a true consequence of locality limitations, and not merely a side-effect of congestion, asynchrony, or limited message-size. As a consequence, a maximum matching approximation is almost as difficult as maximal matching.

In addition, we present two novel algorithms for the distributed approximation of packing and covering problems. The first one is a simple and efficient (also w.r.t. messages sizes and local computations) algorithm for minimum vertex cover (MVC) and maximum matching (MM). In $\mathrm{O}(k)$ rounds, the algorithm computes $\mathrm{O}(\Delta^{1/k})$-approximations for both problems. Thus, in order to achieve a constant-factor approximation, the algorithm needs $\mathrm{O}(\log\Delta)$ rounds. This gives upper bounds which almost match the respective lower bounds, showing that the established time-approximation trade-off is close to optimum.

The second algorithm is based on a network decomposition algorithm by Linial and Saks [13] and works for general packing and covering linear programs. In $\mathrm{O}(k)$ rounds, the algorithm produces an $\mathrm{O}(n^{1/k})$-approximation for all packing and covering LPs. Consequently, in $\mathrm{O}(\log n)$ rounds, a constant-factor approximation can be computed. The algorithm works in the same model which is used for the lower bounds. Therefore, although messages can be large, it proves a strong upper bound for the distributed approximation of general packing and covering problems which is tight up to a factor $\mathrm{O}(\sqrt{\log n\log\log n})$.

The paper is organized as follows. Section 2 summarizes related work and in Section 3 the model of computation is defined. We derive the matching lower bound in Section 4. Section 5 contains an almost tight upper bound. In subsequent Section 6, we give a distributed algorithm for general positive LPs and finally, Section 7 concludes the paper.

2 Related Work

The issue of locality in distributed computing has been of great interest for a long time. Besides being fundamental when designing fast distributed algorithms [18], locality has been exploited for improving fault-tolerance and the local detection of illegal global configurations [1].

There is a great variety of papers studying local algorithms. Most notable in the context of this work are algorithms which compute approximations for packing and covering problems. The distributed approximation of packing and covering LPs based on local information only, has been started by Papadimitriou and Yannakakis [17]. In the model of [17], distributed agents have to solve a given LP without communicating with each other. The first distributed algorithm achieving a constant approximation in a polylogarithmic number of rounds is found in [2].

[3] A maximal matching is a matching which results from a greedy edge-picking process; a maximal matching cannot be improved by adding additional edges to the matching.

[4] Each edge in the maximal matching prevents at most two edges in the optimal maximum matching.

The algorithm of [2] needs $\log^3 n$ rounds in order to achieve a constant approximation. This has recently been improved to $\log^2 n$ in [10]. Distributed algorithms targeted for specific covering and packing problems include algorithms for the minimum dominating set problem [4, 7, 9] as well as algorithms for maximum (weighted) matching [6, 19].

Most of the described algorithms have a time complexity which is at least logarithmic. Not many have studied what happens if the number of rounds is restricted to a constant. Of the above, exceptions are found in [9, 10, 17]. We would also like to mention the influential work of Naor and Stockmayer who show that there are locally checkable labelings which can be computed in distributed constant time [16].

Closely related to packing and covering are maximal independent sets and matchings. Maximal independent sets and maximal matchings can be found in logarithmic time by simple and elegant randomized algorithms [14, 6]. For rings and rooted trees, a maximal independent set can even be constructed in only $\mathrm{O}(\log^* n)$ rounds [3].

However, in spite of the rich literature in the field, not much is known about the fundamental limitations of locality-based approaches. Fich and Ruppert describe a whole bunch of lower bounds and impossibility results in distributed computing [5]. However most of them apply to other computational models where locality is no issue or there are additional more restrictive limiting factors. There have been virtually no nontrivial lower bounds for local computation, besides Linial's pioneering $\mathrm{O}(\log^* n)$ time lower bound for constructing a maximal independent set on a ring [12]. Recently, we have shown that minimum vertex cover and thus covering problems cannot be approximated better than $\Omega(n^{c/k^2}/k)$ and $\Omega(\Delta^{1/k}/k)$ in k communication rounds [8].

3 Model

As mentioned in the previous section, we consider the classic *message passing* model in which the network is modeled by a graph $G = (V, E)$. We assume that in each communication round, each node of the network graph can send an arbitrarily long message to each of its neighbors. Local computations are for free. Each node has a unique identifier and initially, nodes have no knowledge about the network graph. Note that this is the strongest possible model when proving lower bounds on local computation since it focuses entirely on locality and does not consider other aspects arising in the design of distributed computation (unbounded message size, free local computation, ...).

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. The labeling (i.e. the assignment of IDs) of $\mathcal{T}_{v,k}$ is denoted by $\mathcal{L}(\mathcal{T}_{v,k})$. For randomized algorithms, we use the standard trick of determining all random bits at the beginning of the algorithm. In this case, a node v additionally knows about the random bits $\mathcal{R}(\mathcal{T}_{v,k})$ of all nodes in $\mathcal{T}_{v,k}$. The *view* of an edge $e = (u, v)$ is the union of views of its incident nodes

$$\begin{aligned}\mathcal{V}_{e,k} \quad := \quad & (\mathcal{T}_{u,k}, \mathcal{L}(\mathcal{T}_{u,k}), \mathcal{R}(\mathcal{T}_{u,k})) \quad \cup \\ & (\mathcal{T}_{v,k}, \mathcal{L}(\mathcal{T}_{v,k}), \mathcal{R}(\mathcal{T}_{v,k})).\end{aligned}$$

Because message size does not matter, the best an algorithm can do in time k, is to collect an edge's k-neighborhood and base its decision on $\mathcal{V}_{e,k}$. Hence, a randomized algorithm is a deterministic function which assigns a valid output value to each view $\mathcal{V}_{e,k}$.

The model presented is standard and has for example been used in [12, 16] and in textbooks [18].

4 Lower Bound

4.1 Overview

In this section, we derive time lower bounds for distributed maximum matching. More precisely, we prove lower bounds for the more general fractional maximum matching problem (FMM). Let E_i denote the set of edges incident to node v_i. FMM is the natural LP relaxation of MM and defined as

$$\begin{aligned}\max \quad & \sum_{e_j \in E} y_j & & \\ \text{subject to} \quad & \sum_{e_j \in E_i} y_j \leq 1 & \forall v_i \in V & \\ & y_j \geq 0 & \forall e_j \in E. & \end{aligned} \qquad \text{(FMM)}$$

Before dealing with the details of the proof, we give a broad, intuitive outline. As stated in the previous section, the outcome of an edge's decision in a k-local computation is based entirely on the information gathered within its k-neighborhood $\mathcal{V}_{e,k}$. The underlying idea for the lower bound is to construct a graph family G_k in which, for a large number of edges, the information available in $\mathcal{V}_{e,k}$ is not sufficient to distinguish between adjacent edges within k communication rounds. This yields suboptimal local decisions and hence, a suboptimal approximation.

Particularly, we create a graph $G_k = (V, E)$ which contains a subgraph $S = (S_0 \cup S_1, E')$. Each node $v_0 \in S_0$ has δ neighbors in S_1, and each node $v_1 \in S_1$ has δ^2 neighbors in S_0, such that $|S_0| = |S_1| \cdot \delta$. Additionally, each node has exactly one neighbor in its own set of nodes. It follows that $|E'| = \frac{1}{2}(|S_0| + |S_1|) + \delta|S_0|$. By showing that all nodes in S see the same local topology $\mathcal{T}_{v,k}$ within distance k, we will prove the claim that all edges $e \in E'$ have the same distribution of $\mathcal{V}_{e,k}$. This implies that the expected value $\mathrm{E}[y_j]$ of the fractional value y_j must be equal for all edges $e_j \in E'$. Because $|S_0| \gg |S_1|$, an optimal matching consists of all edges within S_0, leading to a matching with cardinality $S_0/2$. In expectation, on the other hand, every algorithm will assign equal fractional values y_j to the edges in E'. In order to obtain a feasible solution, $\mathrm{E}[y_j]$ is bounded by $\mathrm{E}[y_j] \leq 1/\delta^2$, because each node in S_1 has δ^2 such incident edges. By linearity of expectation, the fractional matching in S is of size at most $|E'|/\delta^2 \in \mathrm{O}(|S_0|/\delta)$.

In the sequel, S_0, S_1, and E' denote the sets as described above. The construction of G_k is a two step process. First, the general structure of G_k is defined using the concept of a *cluster-graph* CG_k in Section 4.2. Secondly, in Section 4.3 we construct an instance of G_k obeying the properties imposed by CG_k. Section 4.4 proves that all edges in E' have the same view $\mathcal{V}_{e,k}$ and finally, Section 4.5 derives the lower bounds for FMM.

4.2 The Cluster Graph

We construct the graph $G_k = (V, E)$ featuring the properties described in the previous section. The nodes $v \in V$ are grouped into disjoint sets which are linked to each other as bipartite graphs. For the purpose of describing the structural properties of G_k, we will use a directed *cluster graph* $CG_k = (\mathcal{C}, \mathcal{A})$ with doubly labeled arcs $\ell : \mathcal{A} \to \mathbb{N} \times \mathbb{N}$.. A node $C \in \mathcal{C}$ represents a *cluster*, i.e. one of the disjoint sets, of nodes in G_k. 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 in which each node $u \in C$ has degree δ^c and each node $v \in D$ has degree δ^d. It follows that $|C| \cdot \delta^c = |D| \cdot \delta^d$.

The cluster graph consists of two equal subgraphs, so-called *cluster-trees* CT_k as defined in [8]. In CG_k, we additionally add an arc $\ell(C_i, C_i') := (1, 1)$ between two corresponding nodes of the two cluster trees. Formally, CT_k and CG_k are defined as follows. For the first definition, we call clusters adjacent to exactly one other cluster *leaf-clusters*, and all other clusters *inner-clusters*.

Definition 4.1. *[8] For a given δ and a positive integer k, the **cluster tree** CT_k is recursively defined as follows:*

$$
\begin{aligned}
CT_1 &:= (\mathcal{C}_1, \mathcal{A}_1), \quad \mathcal{C}_1 := \{C_0, C_1, C_2, C_3\} \\
\mathcal{A}_1 &:= \{(C_0, C_1), (C_0, C_2), (C_1, C_3)\} \\
\ell(C_0, C_1) &:= (\delta, \delta^2), \quad \ell(C_0, C_2) := (\delta^2, \delta^3), \\
\ell(C_1, C_3) &:= (\delta, \delta^2)
\end{aligned}
$$

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 $\ell(C_i, C_i') := (\delta^{k+1}, \delta^{k+2})$.*
- *For each leaf-cluster C_i with $(C_p, C_i) \in \mathcal{A}$ and $\ell(C_p, C_i) = (\delta^p, \delta^{p+1})$, add new leaf-clusters C_j' with $\ell(C_i, C_j') := (\delta^j, \delta^{j+1})$ for $j = 1 \ldots k+1, j \neq p+1$.*

Definition 4.2. *Let T_k and T_k' be two instances of CT_k. Further, let C_i and C_i' be corresponding clusters in T_k and T_k', respectively. We obtain the **cluster graph** CG_k by adding an arc $\ell(C_i, C_i') := (1, 1)$ for all clusters $C_i \in CT_k$. Further, we define $n_0 := |C_0 \cup C_0'|$. This uniquely defines the size of all clusters.*

Figure 1 shows CT_2. The shaded subgraph corresponds to CT_1. Figure 2 depicts CG_2, the dashed

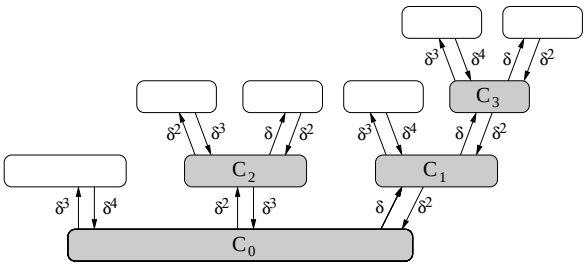

Figure 1: Cluster-Tree CT_2.

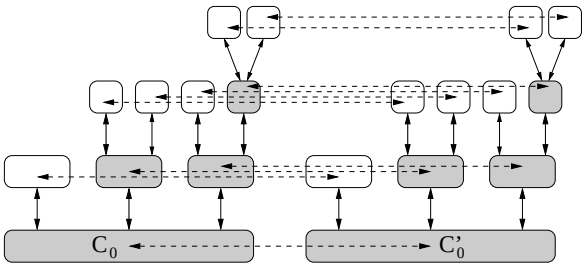

Figure 2: Cluster-Graph CG_2.

lines representing the links $\ell(C_i, C_i') := (1, 1)$. Note that neither CT_k nor CG_k define the adjacency on the level of nodes. They merely prescribe for each node the number of neighbors in each cluster. In the following, we define $S_i := C_i \cup C_i'$ and hence, the sets S_0 and S_1 as referred to in Section 4.1 correspond to $C_0 \cup C_0'$ and $C_1 \cup C_1'$, respectively. The *layer* of a cluster is the distance to C_0 in the cluster tree. We write T_k and T_k' to denote the two cluster trees which constitute CG_k. Further, paying tribute to Figure 2, we call the arcs $\ell(C_i, C_i') := (1, 1)$ *horizontal links* and all other arcs *vertical links*.

4.3 The Lower Bound Graph $\mathbf{G_k}$

Having defined the cluster graph CG_k, it is now our goal to obtain a realization of G_k which has the structure imposed by CG_k and features the additional property that there are no short cycles. As we must prove that the topologies seen by nodes in S_0 and S_1 are identical, the absence of short cycles is of great help. Particularly, if there are no cycles of length $2k + 1$ and less, all nodes see a tree, and we can neglect the difficulty that nodes may see different cycles. The *girth* of a graph G, denoted by $g(G)$, is the length of the shortest cycle in G. Lemma 4.1 states that it is indeed possible to construct G_k as described above.

Lemma 4.1. *If $k + 1 \leq \delta/2$, G_k can be constructed such that the following conditions hold:*

1. *G_k follows the structure of CG_k.*
2. *The girth of G_k is at least $g(G_k) \geq 2k + 1$.*
3. *G_k has $n \leq 4^{2k}\delta^{4k^2}$ nodes.*

Proof. See Appendix A.1. □

4.4 Equality of Views

In this section, we want to prove our claim that all nodes in S_0 and S_1 have the same view and consequently, all edges in E' see the same topology. This task is greatly facilitated since we can use the following result from [8]. For the rather intricate proof, we refer to the original paper.

Lemma 4.2. *[8] Let $\mathcal{G}_k$ be an instance of a cluster tree CT_k with girth $g(\mathcal{G}_k) \geq 2k + 1$. The views of all nodes in clusters C_0 and C_1 are identical up to distance k.*

Because G_k has girth at least $2k+1$ by Lemma 4.1, the two cluster-trees T_k and T_k' constituting G_k must have girth $2k + 1$ as well. It follows from Lemma 4.2 that the desired *equality of views* holds for both T_k and T_k'. Based on this fact, we now show that equality of views holds in G_k, too.

Lemma 4.3. *Let G_k be an instance of a cluster graph CG_k with girth $g(G_k) \geq 2k + 1$. The views of all nodes in clusters S_0 and S_1 are identical up to distance k.*

Proof. Let the *view-tree* $\mathcal{VT}_v$ of a node v be the view of v in CG_k. Particularly, let $\mathcal{VT}_0$ and $\mathcal{VT}_1$ be the view-trees of nodes $v_0 \in S_0$ and $v_1 \in S_1$, respectively. Now, consider a path of length k, $P = (\delta_1, \delta_2, \ldots, \delta_r, \ldots, \delta_k)$ and let δ_r be its first horizontal link. Let $\mathcal{R}_0$ and $\mathcal{R}_1$ be the subtrees of $\mathcal{VT}_0$ and $\mathcal{VT}_1$, seen after following the first $r - 1$ hops of P. By Lemma 4.2, these subtrees are equal with regard to all paths which do never use a horizontal link. By the definition of CG_k, there is such a link

$\ell(C_i, C_i')$ in both $\mathcal{R}_0$ and $\mathcal{R}_1$. Hence, we can write $\mathcal{R}_0 = \beta_0 \cup B_0$ and $\mathcal{R}_1 = \beta_1 \cup B_1$ where β_i denotes the subtrees seen upon using a vertical link and B_i the subtree seen upon using the horizontal link.

By Lemma 4.2, taking a link in B_i does not violate equality. Because the graph is completely symmetric with regard to the subgraphs T_k and T_k' and because the link $\ell(C_i, C_i')$ always connects two equal clusters, the subtrees $\mathcal{R}_i$ and B_i are equivalent. In other words, by taking a horizontal link, we can never create a difference in the view-trees. The remaining such steps easily follow by induction. □

4.5 Analysis

We now derive the lower bounds on the approximation ratio for k-local FMM algorithms. Let OPT be an optimal solution for FMM and let ALG be the solution computed by any algorithm. The main observation is that all nodes in S_0 and S_1 have the same view and therefore, every edge in E' sees the same topology $\mathcal{V}_{e,k}$. This leads to the sub-optimality explained in Section 4.1.

Lemma 4.4. *When applied to $G_k = (V, E)$ as constructed in Subsection 4.3, any distributed, possibly randomized algorithm which runs for at most k rounds computes, in expectation, a solution of at most $ALG \leq |S_0|/(2\delta^2) + (|V| - |S_0|)$.*

Proof. The fractional value assigned to $e_i = (u, v)$ by an algorithm is denoted by y_i, Y_i is the random variable describing the distribution of y_i. The decision of which value y_i is assigned to edge e_i depends only on the view $\mathcal{V}_{e_i,k}$, containing the topologies $\mathcal{T}_{u,k}$ and $\mathcal{T}_{v,k}$, the labelings $\mathcal{L}(\mathcal{T}_{u,k})$ and $\mathcal{L}(\mathcal{T}_{v,k})$ and the randomness $\mathcal{R}(\mathcal{T}_{u,k})$ and $\mathcal{R}(\mathcal{T}_{v,k})$, which e_i can collect during the k communication rounds. Assume an adversary which cooses the labels of the nodes. One possible adversarial strategy is to choose the labeling in G_k uniformly at random. In this case, the labeling $\mathcal{L}(\mathcal{T}_{u,k})$ for an arbitrary node u is chosen uniformly at random, too.

We are now looking at the edges connecting nodes in S_0 and S_1. Clearly all of them see the same topology. If the labels are chosen uniformly at random, it follows that the distribution of the views is the same for all those edges. Let $u \in S_1$ be a node of S_1. The node u has δ^2 neighbors in S_0. Therefore, for edges e_i between nodes in S_0 and S_1, by linearity of expectation, $\mathrm{E}[Y_i] \leq 1/\delta^2$ because otherwise there exist labelings for which the calculated solution is not feasible. By Lemma 4.3, edges e_j with both end-points in S_0 have the same view as edges between S_0 and S_1. Hence, also for the value y_j of e_j, $\mathrm{E}[Y_j] \leq 1/\delta^2$ must hold. There are $|S_0|/2$ such edges and therefore the exected total value contributed by edges between two nodes in S_0 is at most $|S_0|/(2\delta^2)$.

All edges which do not connect two nodes in S_0, have one end-point in $V \setminus S_0$. In order to get a feasible solution, the total value of all edges adjacent to a set of nodes V', can be at most $|V'|$. This can for example be seen by looking at the dual problem, a kind of minimum vertex cover where some edges only have one end node. Clearly, taking all nodes of V' (assigning 1 to the respective variables) yields a feasible solution for this vertex cover problem. The claim then follows by LP duality and the lemma by adding up $|S_0|/(2\delta^2)$ and $|V \setminus S_0|$. □

Lemma 4.5. *If $k + 1 < \delta$, the number of nodes n of G_k is*

$$n \leq |S_0| \left(1 + \frac{k+1}{\delta - (k+1)}\right).$$

Proof. See Appendix A.2. □

We are now ready to derive the lower bound. Lemma 4.4 gives an upper bound on the number of nodes chosen by any k-local FMM algorithm. We do not know OPT, but choosing all horizontal edges within S_0 is certainly feasible. Hence, the optimal solution is lower bounded by $|OPT| \geq |S_0|/2$.

Theorem 4.6. *There are graphs G, such that in k communication rounds, every distributed algorithm for FMM on G has approximation ratios at least $\Omega\left(\frac{n^{c/k^2}}{k}\right)$ and $\Omega\left(\frac{\Delta^{1/k}}{k}\right)$ for some constant $c \geq 1/4$, where n and Δ denote the number of nodes and the highest degree in G, respectively.*

Proof. Assuming $k + 1 \leq \delta/2$, we have $n \leq 2n_0$ by Lemma 4.5, where $n_0 = |S_0|$. Using Lemma 4.1, we have $n_0 \leq n \leq 4^{2k}\delta^{4k^2}$. Using Lemmas 4.4 and 4.5

the approximation ratio α is at least

$$\begin{aligned}\alpha &\geq \frac{n_0/2}{\frac{n_0}{2\delta^2}+(n-n_0)} = \frac{\delta^2 n_0}{n_0+2\delta^2(n-n_0)} \\ &\geq \frac{\delta^3-\delta^2(k+1)}{\delta-(k+1)+2\delta^2(k+1)} \geq \frac{\delta}{2(k+2)} \\ &\geq \frac{n_0^{1/(4k^2)}}{4^{1/(2k)}\cdot 2(k+2)} \in \Omega\left(\frac{n^{1/(4k^2)}}{k}\right).\end{aligned}$$

The second lower bound follows from $\Delta = \delta^{k+2}$. □

Theorem 4.7. *In order to obtain a polylogarithmic or constant approximation ratio, every distributed algorithm for FMM 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. The same lower bounds hold for the construction of maximal matchings and maximal independent sets.*

Proof. See Appendix A.3. □

Remark 1: Choosing the degrees in the cluster graph CG_k in a different way, the approximation lower bounds of Theorem 4.6 for k rounds can be slightly improved to $\Omega(n^{c/k^2}-k)$ and $\Omega(\Delta^{1/k}-k)$. However, this does not affect the asymptotic results of Theorem 4.7.

5 Local Maximum Matching

In order to show that the lower bounds derived in Section 4 are almost tight, we give a simple, synchronous distributed algorithm which achieves an approximation ratio of $\mathrm{O}(\Delta^{1/k})$ in $\mathrm{O}(k)$ rounds and requires $\mathrm{O}(\log\Delta)$ rounds for a constant approximation.

FMM is the dual problem of the fractional minimum vertex cover (FMVC) problem defined as

$$\begin{aligned}\min \quad & \sum_{v_i\in V} x_i \\ \text{s.t.} \quad & x_i + x_j \geq 1 \quad \forall (v_i,v_j)\in E \\ & x_i \geq 0 \quad \forall v_i \in V.\end{aligned} \tag{FMVC}$$

Algorithm 1 MVC-FMM-Algorithm

1: $x_i := 0$; $\forall e_j \in E_i : z_j := 0$;
2: **for** $\ell := k-1$ to 0 by -1 **do**
3: $\quad\tilde{\delta}_i := |\{\text{uncovered edges } e\in E_i\}| = |\tilde{E}_i|$;
4: $\quad\tilde{\delta}_i^{(1)} := \max_{i'\in N_i}\tilde{\delta}_{i'}$;
5: $\quad$**if** $\tilde{\delta}_i \geq (\tilde{\delta}_i^{(1)})^{\ell/(\ell+1)}$ **then**
6: $\quad\quad x_i := 1$;
7: $\quad\quad \forall e_j\in\tilde{E}_i : z_j := z_j + 1/\tilde{\delta}_i$;
8: $\quad$**end if**
9: $\quad Z_i := \sum_{e_j\in E_i} z_j$;
10: $\quad$**if** $(x_i = 0)$ and $(Z_i \geq 1)$ **then**
11: $\quad\quad x_i := 1$;
12: $\quad\quad \forall e_j \in E_i : z_j := z_j\cdot(1+1/Z_i)$;
13: $\quad$**end if**
14: **end for**
15: $Z_i := \sum_{e_j\in E_i} z_j$;
16: $\forall e_j = (v_i,v_{i'}) \in E_i : y_j := z_j/\max\{Z_i, Z_{i'}\}$;

Algorithm 1 approximates both FMM and FMVC. The idea is to compute a feasible solution for minimum vertex cover (MVC) and while doing so, distribute the dual values z_j among the incident edges of each node. Note that we consider FMVC as the primal and FMM as the dual problem. Whenever a node v_i sets its x_i to 1, the sum of the incident z_j values is increased by 1 as well. Hence, at the end of each iteration of the main loop, the invariant

$$\sum_{v_i\in V} x_i = \sum_{e_j\in E} z_j$$

holds. Let E_i be the set of incident edges of node v_i. We will show that for all nodes v_i, $\sum_{e_j\in E_i} z_j \leq \alpha$ for $\alpha = 3+\Delta^{1/k}$ and that consequently, dividing all z_j by α yields a feasible solution for FMM. By LP duality, α is an upper bound on the approximation ratio for FMM and FMVC.

We call an edge *covered* if at least one of its endpoints has joined the vertex cover, i.e. the corresponding x_i is set to 1. The set of uncovered edges incident to a node v_i is denoted by $\tilde{E}_i$, and we define $\tilde{\delta}_i := |\tilde{E}_i|$. The maximum $\tilde{\delta}_{i'}$ among all neighbors $v_{i'}$ of v_i is called $\tilde{\delta}_i^{(1)}$. We call $\tilde{\delta}(v_i)$ the *dynamic degree* of v.

Lemma 5.1. *At the beginning of each iteration, we have* $\tilde{\delta}(v_i) \leq \Delta^{(\ell+1)/k}$ *for all* $v_i \in V$.

Proof. The proof is by induction over the main loop's iterations. For $\ell = k-1$, the lemma follows from the definition of Δ. For subsequent iterations, we show that all nodes having $\tilde{\delta}_i \geq \Delta^{\ell/k}$ set $x_i := 1$ in line 6. In the algorithm, all nodes with $\tilde{\delta}_i \geq (\tilde{\delta}_i^{(1)})^{\ell/(\ell+1)}$ set $x_i := 1$. Hence, we have to show that $\forall i : (\tilde{\delta}_i^{(1)})^{\ell/(\ell+1)} \leq \Delta^{\ell/k}$. By the induction hypothesis, we know that $\tilde{\delta}_i \leq \Delta^{(\ell+1)/k}$ at the beginning of the loop. Since $\tilde{\delta}_i^{(1)}$ represents $\tilde{\delta}_{i'}$ of some node $v_{i'}$, we have $\forall i : \tilde{\delta}_i^{(1)} \leq \Delta^{(\ell+1)/k}$ and the claim follows because $(\tilde{\delta}_i^{(1)})^{\ell/(\ell+1)} \leq \Delta^{\frac{\ell+1}{k}\cdot\frac{\ell}{\ell+1}}$. □

The following lemma bounds the sum of z values in E_i for an arbitrary node v_i. For that purpose, we define $Z_i := \sum_{e_j \in E_i} z_j$.

Lemma 5.2. *At the end of the algorithm, for all nodes $v_i \in V$, $Z_i = \sum_{e_j \in E_i} z_j \leq 3 + \Delta^{1/k}$.*

Proof. Let Φ_h denote the iteration in which $\ell = h$. First, consider a node v_i which does not join the vertex cover. Until Φ_0, we have $Z_i < 1$ since otherwise, it would have set $x_i := 1$ in line 11 of a previous iteration. In Φ_0, $\tilde{\delta}_i = 0$ because all nodes with $\tilde{\delta}_i \geq 1$ set $x_i := 1$ in Φ_0. Therefore, all adjacent nodes $v_{i'}$ have set $x_{i'} := 1$ before Φ_0 and Z_i does not change anymore. Hence, $Z_i < 1$ for nodes which do not belong to the vertex cover constructed by the algorithm.

Next, we consider a node v_i joining the vertex cover in line 6 of an arbitrary Φ_ℓ. With the same argument as above, we know that $Z_i < 1$ at the beginning of Φ_ℓ. Since v_i sets $x_i := 1$, Z_i increases by one. In the same iteration, neighboring nodes $v_{i'}$ may also join the vertex cover and thereby increase Z_i. By the condition in line 5, those nodes have $\tilde{\delta}_{i'} \geq (\tilde{\delta}_{i'}^{(1)})^{\ell/(\ell+1)} \geq \tilde{\delta}_i^{\ell/(\ell+1)}$. Further, by Lemma 5.1, $\tilde{\delta}_i \leq \Delta^{(\ell+1)/k}$ and therefore

$$\frac{\tilde{\delta}_i}{\tilde{\delta}_{i'}} \leq \frac{\tilde{\delta}_i}{\tilde{\delta}_i^{\ell/(\ell+1)}} \leq \tilde{\delta}_i^{1/(\ell+1)} \leq \Delta^{1/k}.$$

Thus, edges which are also covered by a neighboring node $v_{i'}$ get additional increase of the z-value which is at most by a factor $\Delta^{1/k}$ larger. The increase of Z_i in line 6 of Φ_ℓ is then at most $1 + \Delta^{1/k}$. In line 6, the values are distributed only among uncovered edges. Hence, the only way Z_i increases in subsequent iterations is when neighboring nodes set $x_i := 1$ in line 11. The sum of the z_j of all those edges which are covered only by v_i (and are therefore eligible to be increased in this way) is at most 1. In line 12, these z_j can be at most doubled. Putting all together, we have $Z_i \leq 3 + \Delta^{1/k}$ for nodes joining the vertex cover in line 6.

For the last case, we analyze nodes v_i joining the vertex cover in line 11 of Φ_ℓ. Again, we know that $a(v_i) < 1$ at the beginning of Φ_ℓ. Further, using an analogous argument as above, Z_i is increased by at most $\Delta^{1/k}$ due to neighboring nodes joining the vertex cover in line 6 of Φ_ℓ. Through the joining of v_i, Z_i is further increased by 1. Because the z_j are increased proportionally, no further increase of Z_i is possible. Thus, in this case we have $Z_i \leq 2 + \Delta^{1/k}$. □

Theorem 5.3. *The MVC-FMM-Algorithm achieves an approximation ratio of $\mathrm{O}(\Delta^{1/k})$ in k communication rounds. In order to obtain a constant approximation, it requires $\mathrm{O}(\log \Delta)$ rounds.*

Proof. We first prove that the algorithm computes feasible solutions for MVC and FMM. For MVC, this is clear because in the last iteration Φ_0 all nodes having $\tilde{\delta}_i \geq 1$ set $x_i := 1$. The y-values form a feasible solution because in line 16, the z_j of each edge e_j is divided by the larger of the Z_i of the two nodes corresponding to e_j. By Lemma 5.2, each z_j is divided by at most $\alpha = 3 + \Delta^{1/k}$ and therefore, the objective functions of the primal and the dual problem differ by at most a factor α. By LP duality, α is a bound on the approximation ratio for both problems. Setting $k = \beta \log \Delta$ for an appropriate constant β leads to a constant approximation ratio and therefore proves the second claim. □

Remark 1: There is a simple distributed randomized rounding protocol which converts the FMM solution into a matching of essentially the same size. Each edge e_j goes into the matching with probability $y_j/3$. If adjacent edges in the matching are removed, we achieve a $27/4 \cdot \alpha$-approximation where α is the approximation ratio achieved for FMM.

Remark 2: For polylogarithmic approximations, our lower bounds for FMM and MVC are tight

because with the given algorithm, a polylogarithmic approximation ratio can be achieved in $O(\log \Delta / \log\log \Delta)$ rounds.

6 Fast Distributed LP Algorithm

In the previous section, we presented a distributed algorithm to efficiently approximate maximum matching and minimum vertex cover, two classic combinatorial optimization problems. As illustrated, these problems can be formulated as positive LPs. In this section, we broaden our view and give a fast algorithm which computes a solution to general positive LPs of the form

$$\begin{aligned} \min \quad & \underline{c}^{\mathrm{T}} \underline{x} \\ \text{subject to} \quad & A \cdot \underline{x} \geq \underline{b} \\ & \underline{x} \geq \underline{0}. \end{aligned} \tag{LP}$$

where all a_{ij}, b_i and c_i are non-negative. The dual LP for (LP) has the form

$$\begin{aligned} \max \quad & \underline{b}^{\mathrm{T}} \underline{y} \\ \text{subject to} \quad & A^{\mathrm{T}} \cdot \underline{y} \leq \underline{c} \\ & \underline{y} \geq \underline{0}. \end{aligned} \tag{DLP}$$

Let the number of the primal and dual variables be n and m, respectively. Analogously to [2, 17], we consider the following distributed setting. The linear program is bound to a network graph $G = (V, E)$. Each primal variable x_i and each dual variable y_j is associated with a node $v_i^{(p)} \in V$ and $v_j^{(d)} \in V$, respectively. There are communication links between primal and dual nodes wherever the respective variables occur in the corresponding inequality. Thus, $(v_i^{(p)}, v_j^{(d)}) \in E$ if and only if x_i occurs in the j^{th} inequality of (LP). Formally, this means that $v_i^{(p)}$ and $v_j^{(d)}$ are connected if and only if $a_{ji} > 0$.[5] Again, we assume a purely synchronous communication model, but the same approximation ratio can be achieved in an asynchronous environment at the cost of higher message complexity.

[5]Note that in order to solve such a problem in a real network setting where only primal values correspond to nodes, the dual variables may be simulated by the nodes as well.

In [13], Linial and Saks presented a randomized distributed algorithm to decompose a graph into subgraphs of limited diameter. We use their algorithm to decompose the linear program into sub-programs which can be solved locally. Intuitively, the output of the algorithms are connected components of G with the following properties.

(I) Different components are far enough from each other such that we can define a local linear program for each component in a way in which the LPs of any two components do not interfere.

(II) Each node belongs to one of the components with probability at least p, where p depends on the diameter we allow the components to have.

Because of the limited diameter, the LPs of each component can then be computed locally. We apply the decomposition process in parallel often enough such that w.h.p. each node has been selected a logarithmic number of times.

For the decomposition of (LP) and (DLP), we need the following lemma.

Lemma 6.1. *Let $\{x'_1, \ldots, x'_{n'}\}$ be a subset of the primal variables of* (LP) *and let $y'_1, \ldots, y'_{m'}$ be the dual variables which are adjacent to the given subset of the primal variables. Further let* LP′ *and* DLP′ *be LPs where the matrix A' consists only of the columns and rows corresponding to the variables in $\underline{x}'$ and $\underline{y}'$. Every feasible solution for* LP′ *makes the corresponding primal inequalities in* (LP) *feasible and every feasible solution for* DLP′ *is feasible for* (DLP) *(variables not occurring in* LP′ *and* DLP′ *are set to 0). Further, the values of the objective functions for the optimal solutions of* LP′ *and* DLP′ *are smaller than the the optimal values for* (LP) *and* (DLP).

Proof. The feasibilities directly follow from the definition of LP′ and DLP′. The optimal values for the objective functions of LP′ and DLP′ are smaller than the optimal values for (LP) and (DLP) because of the (DLP)-feasibility of a dual feasible solution for DLP′. □

We call LP′ and DLP′ the sub-LPs induced by the subset $\{x'_1, \ldots, x'_{n'}\}$ of primal variables. We apply

the graph decomposition algorithm of [13] to obtain LP$'$ and DLP$'$ (as in Lemma 6.1) which can be solved locally. For a general graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n nodes, the algorithm yields a subset $\mathcal{S} \subseteq \mathcal{V}$ of $\mathcal{V}$ such that each node $u \in \mathcal{S}$ has a leader $\ell(u) \in \mathcal{V}$ and such that the following properties hold.[6]

(I) $\forall u \in \mathcal{S} : \mathrm{d}(u, \ell(u)) < k$

(II) $\forall u, v \in \mathcal{S} : \ell(u) \neq \ell(v) \longrightarrow (u, v) \notin \mathcal{E}$.

(III) $\mathcal{S}$ can be computed in k rounds.

(IV) $\forall u \in \mathcal{V} : \mathrm{P}[u \in \mathcal{S}] \geq \frac{1}{en^{1/k}}$.

$d(u, v)$ denotes the distance between two nodes u and v on $\mathcal{G}$. For the decomposition of the linear program, we define $\mathcal{G}$ such that the node set $\mathcal{V}$ is the set of primal nodes of the graph G and the edge set $\mathcal{E}$ is

$$\mathcal{E} := \left\{(u, v) \,\middle|\, u, v \in \mathcal{V} \wedge \mathrm{d}_G(u, v) \leq 4\right\}.$$

By this, we can guarantee that non-adjacent nodes in $\mathcal{G}$ do not have neighboring dual nodes in G whose variables occur in the same constraint of (DLP). Further, a message over an edge of $\mathcal{G}$ can be sent in 4 rounds on the network graph G. The basic algorithm for a primal node v to approximate (LP) and (DLP) then works as follows:

1: Run graph decomposition of [13] on $\mathcal{G}$;
2: **if** $v \in \mathcal{S}$ **then**
3: **send** IDs of dual neighbors to $\ell(v)$.
4: **end if**;
5: **if** $v = \ell(u)$ for some $u \in \mathcal{S}$ **then**
6: compute local LP/DLP (cf. Lemma 6.1) of variables of $u \in \mathcal{S}$ for which $v = \ell(u)$.
7: **send** resulting values to nodes holding the respective variables.
8: **end if**

The dual nodes only forward messages in steps 1, 3, and 7 and receive the values for their variables in step 7. We now have a closer look at the locally computed LPs in line 6. By Property (II) of the graph decomposition algorithm, primal variables belonging to different local LPs cannot occur in the same primal constraint (otherwise, the according primal nodes had to be neighbors in $\mathcal{G}$). The analogous fact holds for dual variables since primal nodes belonging to different local LPs have distance at least 6 on G and thus dual nodes belonging to different local LPs have distance at least 4 on G. Therefore, the local LPs do not interfere and together they form the sub-LPs induced by $\mathcal{S}$ (cf. Lemma 6.1).

The complete LP approximation algorithm now consists of N independent parallel executions of the described basic algorithm. The variables of the N sub-LPs are added up and in the end, primal/dual nodes divide their variables by the maximum/minimum possible value to keep/make all constraints they occur in feasible.[7] The following theorem states how N must chosen such that the obtained approximation ratio is optimized.

Theorem 6.2. *Let $N = \alpha e n^{1/k} \ln n$ for $\alpha \approx 4.51$. Executing the basic algorithm N times, summing up the variables of the N execution and dividing these sums as described, yields an $\alpha e n^{1/k}$ approximation of (LP)/(DLP) w.h.p. The algorithm needs $\mathrm{O}(k)$ rounds to complete.*

Proof. See Appendix A.4 □

Collorary 6.3. *Using the network decomposition algorithm of [13], in only $\mathrm{O}(k)$ rounds (LP) and (DLP) can be approximated by a factor $\mathrm{O}(n^{1/k})$ w.h.p. For $k \in \Theta(\log n)$, this gives a constant factor approximation in $\mathrm{O}(\log n)$ rounds.*

7 Conclusions

The importance of locality stems from the desire and necessity to achieve a global goal based on local information. Algorithms based on local information appear to be the choice at hand for numerous problems related to large-scale distributed systems and new networking fields, such as mobile computing. Unfortunately, with few exceptions [8, 12], the impact of locality on distributed algorithms is not thoroughly understood and we believe that there is an urging need for a solid theoretical foundation on which future work can be built. We hope and believe that the lower bounds (as well as the method) given in the present paper will be a step towards this goal.

[6] We use $p = 1/n^{1/k}$ in the algorithm of Section 4 of [13], the properties then directly follow from Lemma 4.1 of [13].

[7] The primal and dual variables x_i and y_j are divided by $\min_{j \in N_i} \frac{1}{b_j} \sum_\ell a_{j\ell} x_\ell$ and $\max_{i \in N_j} \frac{1}{c_i} \sum_\ell a_{\ell i} y_\ell$, respectively.

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] Y. Bartal, J. W. Byers, and D. Raz. Global Optimization Using Local Information with Applications to Flow Control. In *Proc. of the 38th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 303–312, 1997.

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

[4] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan. Fast Distributed Algorithms for (Weakly) Connected Dominating Sets and Linear-Size Skeletons. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 717–724, 2003.

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

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

[7] L. Jia, R. Rajaraman, and R. Suel. An Efficient Distributed Algorithm for Constructing Small Dominating Sets. In *Proc. of the 20^{th} ACM Symposium on Principles of Distributed Computing (PODC)*, pages 33–42, 2001.

[8] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What Cannot Be Computed Locally! Technical Report 438, ETH Zurich, Dept. of Computer Science, 2004.

[9] F. Kuhn and R. Wattenhofer. Constant-Time Distributed Dominating Set Approximation. In *Proc. of the 22^{nd} 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] 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.

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

[13] N. Linial and M. Saks. Low Diameter Graph Decompositions. *Combinatorica*, 13(4):441–454, 1993.

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

[15] A. Mayer, M. Naor, and L. Stockmeyer. Local Computations on Static and Dynamic Graphs. In *Proc. of IEEE 3^{rd} Israeli Symp. on Theory of Computing and Systems (ISTCS)*, pages 268–278, 1995.

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

[17] C. Papadimitriou and M. Yannakakis. Linear Programming without the Matrix. In *Proc. of the 25th ACM Symposium on Theory of Computing (STOC)*, pages 121–129, 1993.

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

[19] M. Wattenhofer and R. Wattenhofer. Distributed Weighted Matching. Technical Report 420, ETH Zurich, Department of Computer Science, 2003.

A Appendix

A.1 Proof of Lemma 4.1

We give a constructive prove. We want to construct G_k with girth at least $2k+1$. We start by creating an arbitrary intermediate graph $\tilde{G}_k = (\tilde{V}, \tilde{E})$ (with $\tilde{n}_0 = |\tilde{S}_0|$ and $\tilde{n} = |\tilde{V}|$), which may be of minimal girth 4 and then construct G_k from $\tilde{G}_k$.

No node in G_k has more than δ^{k+1} neighbors in a leaf-cluster of CG_k. Hence, we can set the size of the outermost clusters (on layer $k+2$) to δ^{k+1}. Since the number of nodes increases by a factor of δ for each level on the way to S_0 and since there are $k+2$ levels (including S_0), the size of a cluster on level l is δ^{2k+3-l} and consequently, $\tilde{n}_0 = 2\delta^{2k+3}$. We construct $\tilde{G}_k$ as follows: Let $\tilde{C}_i$ and $\tilde{C}_j$ be two adjacent clusters with a vertical link (δ^i, δ^{i+1}). $\tilde{C}_i$ and $\tilde{C}_j$ can be connected by as many complete bipartite graphs $K_{\delta^i,\delta^{i+1}}$ as necessary. As for horizontal links $(1,1)$, we connect the clusters with a $K_{1,1}$ in the obvious way, by connecting two arbitrary nodes from both clusters. Since $|\tilde{C}_i| = |\tilde{C}'_i|$, this is always possible.

The construction of G_k from $\tilde{G}_k$ is based on the construction of the graph family $D(r,q)$ as proposed in [11] and follows a technique already used in [8]. For given r and q, $D(r,q)$ defines a bipartite graph with $2q^r$ nodes and girth $g(D(r,q)) \geq r+5$. We show that for appropriate r and q, an instance of G_k is obtained by deleting some of the edges of $D(r,q)$. Since deleting edges cannot create cycles, $g(G_k) \geq r+5$ follows.

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 labeled by the r-vectors over the finite field $\mathbb{F}_q$, i.e. $P = L = \mathbb{F}_q^r$. In accordance with [11], 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):

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

Note that this admittedly somewhat confusing notation has been chosen in order to facilitate the following system of equations. There is an edge between two nodes (p) and $[l]$, exactly if the first $r-1$ of the following equations hold (for $i = 2, 3, \dots$).

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

As shown in [11], $D(r,q)$ has girth at least $r+5$ for odd $r \geq 3$. We need the following helper lemma which can be shown to be true [8] because the linear system for the unknown coordinates defined by the first $r-1$ equations is a lower triangular matrix of full rank and therefore posseses a unique solution.

Lemma A.1. *[8] 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.*

Observing that both G_k and $\tilde{G}_k$ are bipartite graphs. One partition consists of all odd-layer clusters in T_k and all even-layer clusters in T'_k. The other consists of the remaining clusters. The partitions are of equal size, since the graph is completely symmetric with regard to T_k and T'_k. We choose q to be the smallest prime power greater than or equal to $|\tilde{V}|/2 = \tilde{n}/2$. In both partitions $V_1(\tilde{G}_k)$ and $V_2(\tilde{G}_k)$ of $\tilde{G}_k$, we uniquely label all nodes v with elements $c(v) \in \mathbb{F}_q$.

Now, we choose q as described above and 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 $u \in V_1(\tilde{G}_k)$ and $v \in V_2(\tilde{G}_k)$ for which $c(u) = p_1$ and $c(v) = l_1$. Finally, nodes without incident edges are removed from G_k. By this process, each node in $\tilde{G}_k$ is replaced by q^{2k-5} new nodes in G_k. Further, a cluster C_i consists of all nodes (p) and $[l]$ which have their first coordinates equal the labels of nodes in C_i. Therefore, the graph G_k constructed as described is a cluster graph with

the degrees δ^i as in $\tilde{G}_k$. G_k has $\tilde{n}q^{2k-5}$ nodes and girth at least $2k+1$.

In order to bound the number of nodes, note that because q is the smallest prime power greater than or equal to $\tilde{n}/2$, we have $q \leq \tilde{n}$. The sizes of clusters decreases by a factor δ and the number of clusters goes up by at most a factor $k+1$ per layer. Therefore, if we assume that $k+1 \leq \delta/2$, the number of nodes on layer 0 is at least half of the total number of nodes, i.e. $\tilde{n} \leq 2\tilde{n}_0$. As described above, we have $\tilde{n}_0 = 2\delta^{2k+3}$ and therefore

$$n \;=\; \tilde{n}q^{2k-5} \;\leq\; 4\delta^{2k+3}\left(4\delta^{2k+3}\right)^{2k-5} \;\leq\; 4^{2k}\delta^{4k^2}.$$

This concludes the proof. □

A.2 Proof of Lemma 4.5

The number of nodes per cluster decreases by a factor δ for each layer. A cluster on layer l contains $|S_0|/\delta^l$ nodes. By the definition of CG_k, each cluster has no more than $k+1$ neighboring clusters on a higher layer. Therefore, the number of nodes n_l on layer l is upper bounded by

$$n_l \;\leq\; (k+1)^l \cdot \frac{|S_0|}{\delta^l}.$$

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

$$\begin{aligned} n \;&\leq\; |S_0| \cdot \sum_{i=0}^{k+1}\left(\frac{k+1}{\delta}\right)^l \;\leq\; |S_0| \cdot \sum_{i=0}^{\infty}\left(\frac{k+1}{\delta}\right)^l \\ &=\; |S_0| + |S_0|\left(\frac{k+1}{\delta}\right)\left(\frac{1}{1-\frac{k+1}{\delta}}\right) \\ &=\; |S_0|\left(1+\frac{k+1}{\delta-(k+1)}\right). \end{aligned}$$

□

A.3 Proof of Theorem 4.7

We set $k = \beta\sqrt{\log n/\log\log n}$ for an arbitrary constant $\beta > 0$. Plugging this into the first lower bound of Theorem 4.6, 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 constant hidden 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

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

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 follows.

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

The lower bounds also hold for the construction of maximal matchings because a maximal matching is a 2-approximation for maximum matching. A maximal matching of a graph $G = (V, E)$ corresponds to a maximal independent set on the line graph of G with node set E and edges between nodes corresponding to edges with common end node in G.

□

A.4 Proof of Theorem 6.2

We begin the proof with the running time. The N executions can be performed completely in parallel and we therefore focus on one instance of the basic algorithm. By Property (I), the topology collecting and variable distribution in lines 3 and 7 can be performed in $\mathrm{O}(k)$ time. The same holds for the graph decomposition in line 1 by Property (III).

For the approximation ratio, we have to bound the ratio of the factors by which the primal and the dual variables are divided in the end. By Lemma 6.1, the dual variables of each of the N sub-LPs constitute a feasible solution for (DLP). Therefore, the sums of the dual variables of the sub-LPs have to be divided by at most N to obtain a feasible solution for (DLP). For the primal variables, we have to count the number occurrences in sub-LPs for each primal constraint. This is lower-bounded by the number of times each primal node has been chosen to be in $\mathcal{S}$. By (IV), for each primal node, the probability in each of the N executions is at least $1/(en^{1/k})$. We use

Chernoff bounds to obtain an upper bound on the probability that primal node v occurs in less than $\ln n$ sub-LPs. Let X denote the number of times, v is chosen by the graph decomposition algorithm:

$$\begin{aligned} \mathrm{P}\left[X < \ln n\right] &< \left(\frac{\alpha^{1/\alpha}}{e^{1-1/\alpha}}\right)^{\alpha \ln n} \\ = \frac{e^{\ln \alpha \ln n}}{e^{(\alpha-1)\ln n}} &= \frac{1}{n^{\alpha-1-\ln \alpha}} < \frac{1}{n^2} \end{aligned}$$

for $\alpha > 4.51$. Thus, with probability at least $1 - 1/n$ all primal variables can be divided by $\ln n$. □