# Approximating Bipartite Minimum Vertex Cover in the CONGEST Model

**Salwa Faour**
Albert-Ludwigs-Universität Freiburg, Germany
salwa.faour@cs.uni-freiburg.de

**Fabian Kuhn**
Albert-Ludwigs-Universität Freiburg, Germany
kuhn@cs.uni-freiburg.de

## Abstract

We give efficient distributed algorithms for the minimum vertex cover problem in bipartite graphs in the CONGEST model. From Kőnig's theorem, it is well known that in bipartite graphs the size of a minimum vertex cover is equal to the size of a maximum matching. We first show that together with an existing $O(n \log n)$-round algorithm for computing a maximum matching, the constructive proof of Kőnig's theorem directly leads to a deterministic $O(n \log n)$-round CONGEST algorithm for computing a minimum vertex cover. We then show that by adapting the construction, we can also convert an *approximate* maximum matching into an *approximate* minimum vertex cover. Given a $(1-\delta)$-approximate matching for some $\delta > 1$, we show that a $(1+O(\delta))$-approximate vertex cover can be computed in time $O\big(D + \mathrm{poly}\left(\frac{\log n}{\delta}\right)\big)$, where $D$ is the diameter of the graph. When combining with known graph clustering techniques, for any $\varepsilon \in (0,1]$, this leads to a $\mathrm{poly}\left(\frac{\log n}{\varepsilon}\right)$-time deterministic and also to a slightly faster and simpler randomized $O\big(\frac{\log n}{\varepsilon^3}\big)$-round CONGEST algorithm for computing a $(1+\varepsilon)$-approximate vertex cover in bipartite graphs. For constant $\varepsilon$, the randomized time complexity matches the $\Omega(\log n)$ lower bound for computing a $(1+\varepsilon)$-approximate vertex cover in bipartite graphs even in the LOCAL model. Our results are also in contrast to the situation in general graphs, where it is known that computing an optimal vertex cover requires $\tilde{\Omega}(n^2)$ rounds in the CONGEST model and where it is not even known how to compute any $(2-\varepsilon)$-approximation in time $o(n^2)$.

**2012 ACM Subject Classification** Theory of computation → Distributed algorithms; Networks → Network algorithms

**Keywords and phrases** distributed vertex cover, distributed graph algorithms, distributed optimization, bipartite vertex cover

**Digital Object Identifier** 10.4230/LIPIcs.OPODIS.2020.29

**Related Version** A full version of the paper is available at `https://arxiv.org/abs/2011.10014`.

## 1 Introduction & Related Work

In the minimum vertex cover (MVC) problem, we are given an $n$-node graph $G = (V, E)$ and we are asked to find a vertex cover of smallest possible size, that is, a minimum cardinality subset of $V$ that contains at least one node of every edge in $E$. In the distributed MVC problem, the graph $G$ is the network graph and the nodes of $G$ have to compute a vertex cover by communicating over the edges of $G$. At the end of a distributed vertex cover algorithm, every node $v \in V$ must know if it is contained in the vertex cover or not. Different variants of the MVC problem have been studied extensively in the distributed setting, see e.g., [3, 4, 6, 7, 9, 11, 16, 18–20, 25, 26]. Classically, when studying the distributed MVC problem and also related distributed optimization problems on graphs, the focus has been on understanding the *locality* of the problem. The focus therefore has mostly been


24th International Conference on Principles of Distributed Systems (OPODIS 2020).
Editors: Quentin Bramas, Rotem Oshman, and Paolo Romano; Article No. 29; pp. 29:1–29:16
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

on establishing how many synchronous communication rounds are necessary to solve or approximate the problem in the LOCAL model, that is, if in each round, each node of $G$ can send an arbitrarily large message to each of its neighbors.

**MVC in the LOCAL model.** The minimum vertex cover problem is closely related to the maximum matching problem, i.e., to the problem of finding a maximum cardinality set of pairwise non-adjacent (i.e., disjoint) edges. Since for every matching $M$, any vertex cover has to contain at least one node from each of the edges $\{u, v\} \in M$, the size of a minimum vertex cover is lower bounded by the size of a maximum matching. We therefore obtain a simple 2-approximation $S$ for the MVC problem by first computing a maximal matching and by defining the vertex cover $S$ as $S := \bigcup_{\{u,v\}\in M} \{u, v\}$. It has been known since the 1980s that a maximal matching can be computed in $O(\log n)$ rounds by using a simple randomized algorithm [2, 22, 29]. The fastest known randomized distributed algorithm for computing a maximal matching has a round complexity of $O(\log \Delta + \log^3 \log n)$, where $\Delta$ is the maximum degree of the graph $G$ [8, 15], and the fastest known deterministic algorithm has a round complexity of $O(\log^2 \Delta \cdot \log n)$ [15]. A slightly worse approximation ratio of $2 + \varepsilon$ can even be achieved in time $O\big(\frac{\log \Delta}{\log \log \Delta}\big)$ for any constant $\varepsilon > 0$. This matches the $\Omega\big(\min\big\{\frac{\log \Delta}{\log \log \Delta}, \sqrt{\frac{\log n}{\log \log n}}\big\}\big)$ lower bound of [25], which even holds for any polylogarithmic approximation ratio. In [18], it was further shown that there exists a constant $\varepsilon > 0$ such that computing a $(1+\varepsilon)$-approximate solution for MVC requires $\Omega(\log n)$ rounds even for bipartite graphs of maximum degree 3. By using known randomized distributed graph clustering techniques [27, 30], this bound can be matched: For any $\varepsilon \in (0, 1]$, a $(1 + \varepsilon)$-approximate MVC solution can be computed in time $O\big(\frac{\log n}{\varepsilon}\big)$ in the LOCAL model. It was shown in [17] that in fact all distributed covering and packing problems can be $(1 + \varepsilon)$-approximated in time poly $\big(\frac{\log n}{\varepsilon}\big)$ in the LOCAL model. By combining with the recent deterministic network decomposition algorithm of [32], the same result can even be achieved deterministially. We note that all the distributed $(1 + \varepsilon)$-approximations for MVC and related problems quite heavily exploit the power of the LOCAL model. They use very large messages and also the fact that the nodes can do arbitrary (even exponential-time) computations for free.

**MVC in the CONGEST model.** As the complexity of the distributed minimum vertex cover and related problems in the LOCAL model is now understood quite well, there has recently been increased interest in also understanding the complexity of these problems in the more restrictive CONGEST model, that is, when assuming that in each round, every node can only send an $O(\log n)$-bit message to each of its neighbors. Some of the algorithms that have been developed for the LOCAL model do not make use of large messages and they therefore directly also work in the CONGEST model. This is in particular true for all the maximal matching algorithms and also for the $(2 + \varepsilon)$-approximate MVC algorithm mentioned above. Also in the CONGEST model, it is therefore possible to compute a 2-approximation for MVC in $O(\log \Delta + \log^3 \log n)$ rounds and a $(2 + \varepsilon)$-approximation in $O\big(\frac{\log \Delta}{\log \log \Delta}\big)$ rounds. However, there is no non-trivial (i.e., $o(n^2)$-round) CONGEST MVC algorithm known for obtaining an approximation ratio below 2. For computing an optimal vertex cover on general graphs, it is even known that $\tilde{\Omega}(n^2)$ rounds are necessary in the CONGEST model [11]. It is therefore an interesting open question to investigate if it is possible to approximate MVC within a factor smaller than 2 in the CONGEST model or to understand for which families of graphs, this is possible. The only result in this direction that we are aware of is a recent paper that gives $(1 + \varepsilon)$-approximation for MVC in the square graph $G^2$ in $O(n/\varepsilon)$ CONGEST rounds on the underlying graph $G$ [6].

**MVC in bipartite graphs.** In the present paper, we study the distributed complexity of MVC in the CONGEST model for bipartite graphs. Unlike for general graphs, where MVC is APX-hard (and even hard to approximate within a factor $2-\varepsilon$ when assuming the unique games conjecture [24]), for bipartite graphs, MVC can be solved optimally in polynomial time. While in general graphs, we only know that a minimum vertex cover is at least as large as a maximum matching and at most twice as large as a maximum matching, for bipartite graphs, Kőnig's well-known theorem [12, 23] states that in bipartite graphs, the size of a maximum matching is always equal to the size of a minimum vertex cover. In fact, if one is given a maximum matching of a bipartite graph $G = (U \cup V, E)$, a vertex cover of the same size can be computed in the following simple manner. Assume that we are given the bipartition of the nodes of $G$ into sets $U$ and $V$ and assume that we are given a maximum matching $M$ of $G$. Now, let $L_0 \subseteq U$ be the set of unmatched nodes in $U$ and let $L \subseteq U \cup V$ be the set of nodes that are reachable from $L_0$ over an alternating path (i.e, over a path that alternates between edges in $E \setminus M$ and edges in $M$). It is not hard to show that the set $S := (U \setminus L) \cup (V \cap L)$ is a vertex cover that contains exactly one node of every edge in $M$. We note that this construction also directly leads to a distributed algorithm for computing an optimal vertex cover in bipartite graphs $G$. The bipartition of $G$ can clearly be computed in time $O(D)$, where $D$ is the diameter of $G$ and given a maximum matching $M$, the set $L$ can then be computed in $O(n)$ rounds by doing a parallel BFS exploration on alternating paths starting at all nodes in $L_0$. Together with the $O(n \log n)$-round CONGEST algorithm of [1] for computing a maximum matching, this directly leads to a deterministic $O(n \log n)$-round CONGEST algorithm for computing an optimal vertex cover in bipartite graphs. As our main contribution, we show that it is not only possible to efficiently convert an optimal matching into an optimal vertex cover, but we can also efficiently turn an *approximate* solution of the maximum matching problem in a bipartite graph into an *approximate* solution of the MVC problem on the same graph. Unlike for MVC, where no arbitrarily good approximation algorithms are known for the CONGEST model, such algorithms are known for the maximum matching problem [1, 5, 28]. We use this to develop polylogarithmic-time approximation schemes for the bipartite MVC problem in the CONGEST model. We next discuss our main contributions in more detail.

## 1.1 Contributions

Our first contribution is a simple linear-time algorithm to solve the exact minimum vertex cover problem.

▶ **Theorem 1.** *There is a deterministic* CONGEST *algorithm to (exactly) solve the minimum vertex cover problem in bipartite graphs in time* $O(\mathsf{OPT} \cdot \log \mathsf{OPT})$*, where* OPT *is the size of a minimum vertex cover.*

**Proof.** As mentioned, the algorithm is a straightforward CONGEST implementation of Kőnig's constructive proof. Given a bipartite graph $G = (U \cup V, E)$, one first computes a maximum matching $M$ of $G$ in time $O(\mathsf{OPT} \cdot \log \mathsf{OPT})$ by using the CONGEST algorithm of [1]. One elects a leader node $\ell$ and computes a BFS tree of $G$ rooted at $\ell$ in time $O(D)$, where $D$ is the diameter of $G$. Let $U$ be the set of nodes at even distance from $\ell$ and let $V$ be the set of nodes at odd distance from $\ell$. Let $L_0$ be the set of nodes in $U$ that are not contained in any edge of $M$. Starting at $L_0$, we do a parallel BFS traversal on alternating paths. Let $L$ be the set of nodes that are reached in this way. The set $L$ can clearly be computed in time $O(|M|) = O(\mathsf{OPT})$. As shown in the constructive proof of Kőnig's theorem [12, 23], the minimum vertex cover $S$ is now defined as $S := (U \setminus L) \cup (V \cap L)$. ◀

Our main results are two distributed algorithms to efficiently compute $(1+\varepsilon)$-approximate solutions to the minimum vertex cover problem. We first give a slightly more efficient (and also somewhat simpler) randomized algorithm.

▶ **Theorem 2.** *For every $\varepsilon \in (0,1]$, there is a randomized* CONGEST *algorithm that for any bipartite $n$-node graph $G$ computes a vertex cover of expected size at most $(1+\varepsilon)\cdot \mathsf{OPT}$ in time $O\big(\frac{\log n}{\varepsilon^3}\big)$, w.h.p., where* OPT *is the size of a minimum vertex cover of $G$.*

We remark that for constant $\varepsilon$, the above result matches the lower bound of [18] for the LOCAL model. More precisely, in [18], it is shown that there exists a constant $\varepsilon > 0$ for which computing a $(1+\varepsilon)$-approximation of minimum vertex cover requires $\Omega(\log n)$ rounds even on bounded-degree bipartite graphs. The second main result shows that similar bounds can also be achieved deterministically.

▶ **Theorem 3.** *For every $\varepsilon \in (0,1]$, there is a deterministic* CONGEST *algorithm that for any bipartite $n$-node graph $G$ computes a vertex cover of size at most $(1+\varepsilon)\cdot \mathsf{OPT}$ in time $\mathrm{poly}\big(\frac{\log n}{\varepsilon}\big)$, where* OPT *is the size of a minimum vertex cover of $G$.*

## 1.2 Our Techniques in a Nutshell

We next describe the key ideas that leads to the results in Theorems 2 and 3. The core of our algorithms is a method to efficiently transform an approximate solution $M$ for the maximum matching problem into an approximate solution of MVC. More concretely, assume that we are given a matching $M \subseteq E$ of a bipartite graph $G = (U \cup V, E)$ such that $M$ is a $(1-\varepsilon)$-approximate maximum matching of $G$ (for a sufficiently small $\varepsilon > 0$). In Section 3, we then first show that we can compute a vertex cover $S \subseteq U \cup V$ of size $(1+O(\varepsilon\,\mathrm{poly}\log n))\cdot|M|$ (and therefore a $(1+O(\varepsilon\,\mathrm{poly}\log n))$-approximation for MVC) in time $O\big(D + \mathrm{poly}\big(\frac{\log n}{\varepsilon}\big)\big)$, where $D$ is the diameter of $G$. If the matching $M$ has the additional property that there are no augmenting paths of length at most $2k-1$ for some $k = O(1/\varepsilon)$, we show that such a vertex cover $S$ can be obtained by adapting the constructive proof of Kőnig's theorem. Clearly, the bipartition of the nodes of $G$ into sets $U$ and $V$ can be computed in time $O(D)$. Now, we again define $L_0$ as the set of unmatched nodes in $U$ and more generally for any integer $i \in \{1,\dots,2k\}$, we define $L_i$ to be the set of nodes in $U \cup V$ that can be reached over an alternating path of length $i$ from $L_0$ and for which no shorter such alternating path exists. Note that all nodes in set $L_{2j-1}$ for $j \in \{1,\dots,k\}$ are matched nodes as otherwise, we would have an augmenting path of length at most $2k-1$. Note that any alternating path starting at $L_0$ starts with a non-matching edge from $U$ to $V$ and it alternates between non-matching edges from $U$ to $V$ and matching edges from $V$ to $U$. For every $j \geq 1$, the set $L_{2j}$ therefore exactly contains the matching neighbors of the nodes in $L_{2j-1}$ and we therefore have $|L_{2j}| = |L_{2j-1}|$. We will show that for every $j \in \{1,\dots,k\}$ the set

$$S_j := \bigcup_{j' \in \{1,\dots,j\}} L_{2j'-1} \cup \left( U \setminus \bigcup_{j' \in \{0,\dots,j-1\}} L_{2j'} \right)$$

is a vertex cover of size $|M| + |L_{2j}| = |M| + |L_{2j-1}|$. Because the sets $L_i$ are disjoint, clearly one of these vertex covers must have size at most $\big(1+\frac{1}{k}\big)\cdot|M| = (1+O(\varepsilon))\cdot|M|$.

If we do not have the guarantee that $M$ does not have short augmenting paths, we show that one can first delete $O(\varepsilon\cdot|M|\cdot\mathrm{poly}\log n)$ nodes from $U \cup V$ such that in the induced subgraph of the remaining nodes, there are no short augmenting paths w.r.t. $M$. We also show that we can find such a set of nodes to delete in time poly $\left(\frac{\log n}{\varepsilon}\right)$. We can therefore then first compute a good vertex cover approximation for the remaining graph and we then obtain a vertex cover of $G$ by also adding all the removed nodes to the vertex cover.

Given our algorithm to compute a good MVC approximation in time $O(D + \text{poly}\log n)$ in Section 4, we show how that in combination with known graph clustering techniques, we can obtain MVC approximation algorithms with polylogarithmic time complexities and thus prove Theorems 2 and 3. Given a maximal matching $M$, we show that we can compute disjoint low-diameter clusters such that all the edges between clusters can be covered by $O(\varepsilon \cdot |M|)$ nodes. With randomization, such a clustering can be computed by using the random shifts approach of [10, 30] and deterministically such a clustering can be computed by a simple adaptation of the recent network decomposition algorithm of [32]. Since the clusters have a small diameter, we can then use the algorithm of Section 3 described above inside the clusters to efficiently compute a good MVC approximation.

## 2 Model and Definitions

**Communication Model.** We work with the standard CONGEST model [31]. The network is modelled as an $n$-node undirected graph $G = (V, E)$ with maximum degree at most $\Delta$ and each node has a unique $O(\log n)$-bit identifier. The computation proceeds in synchronous communication rounds. Per round, each node can perform some local computations and send one $O(\log n)$-bit message to each of its neighbors. At the end, each node should know its own part of the output, e.g., whether it belongs to a vertex cover or not.

**Low-Diameter Clustering.** In order to reduce the problem of approximating MVC on general (bipartite) graphs to approximating MVC on low-diameter (bipartite) graphs, we need a slightly generalized form of a standard type of graph clustering. Let $G = (V, E, w)$ be a weighted graph with non-negative edge weights $w(e)$ and assume that $W := \sum_{w \in E} w(e)$ is the total weight of all edges in $G$. A subset $S \subseteq V$ of the nodes of $G$ is called $\lambda$-*dense* for $\lambda \in [0, 1]$ if the total weight of the edges of the induced subgraph $G[S]$ is at least $\lambda \cdot W$. A *clustering* of $G$ is a collection $\{S_1, \ldots, S_k\}$ of disjoint subsets $S_i \subseteq V$ of the nodes. A clustering $\{S_1, \ldots, S_k\}$ is called $\lambda$-dense if the set $S := S_1 \cup \cdots \cup S_k$ is $\lambda$-dense. The *strong diameter* of a cluster $S_i \subseteq V$ is the (unweighted) diameter of the induced subgraph $G[S_i]$ and the *weak diameter* of a cluster $S_i \subseteq V$ is the maximum (unweighted) distance in $G$ between any two nodes in $S_i$. The strong/weak diameter of a clustering $\{S_1, \ldots, S_k\}$ is the maximum strong/weak diameter of any cluster $S_i$. A clustering $\{S_1, \ldots, S_k\}$ is called $h$-*hop separated* for some integer $h \geq 1$ if for any two clusters $S_i$ and $S_j$ $(i \neq j)$, we have $\min_{(u,v) \in S_i \times S_j} d_G(u, v) \geq h$, where $d_G(u, v)$ denotes the hop-distance between $u$ and $v$ in $G$. A clustering $\{S_1, \ldots, S_k\}$ is called $(c, d)$-*routable* if we are in addition given a collection of trees $T_1, \ldots, T_k$ in $G$ such that for every $i \in \{1, \ldots, k\}$, the node set of $T_i$ contains the nodes in $S_i$, the height of $T_i$ is at most $d$ and every edge $e \in E$ of $G$ is contained in at most $c$ trees $T_1, \ldots, T_k$. Note that a $(c, d)$-routable clustering clearly has weak diameter at most $2d$. Note also that any clustering with strong diameter $d$ can easily be extended to a $(1, d)$-routable clustering by computing a BFS tree $T_i$ for the induced subgraph $G[S_i]$ of each cluster $S_i$.

## 3 Approximating MVC in Time Linear in the Diameter

In this section, we show how to compute a minimum vertex cover approximation in time $O(D + \text{poly}\log n)$ in the CONGEST model, where $D$ is the diameter of the graph. Before discussing a distributed algorithm, we first describe a generic high-level algorithm to compute a $(1-\varepsilon)$-approximate vertex cover from an appropriate approximate matching $M$ of a bipartite graph $G$. Given a matching $M$ of any graph $G$, a path is said to be augmenting w.r.t. $M$ in $G$ if it is a path that starts and ends with unmatched vertices and alternates between matched and unmatched edges. Inspired by the standard constructive proof of Kőnig's theorem, we first describe an algorithm that gives an approximate minimum vertex cover in bipartite graphs from an approximate maximum matching with the guarantee that no short augmenting paths exist in the graph. We remark that a similar construction has also been used by Feige, Mansour, and Schapire for approximating the bipartite MVC problem in the local computation algorithms model [14].

In the following, assume that $G=(V,E)$ is a bipartite graph, where the bipartition of $V$ is given by $V=A\cup B$. Let $k\geq 1$ be an integer parameter and assume that $M$ is a matching of $G$ with no augmenting paths of length $2k-1$ or shorter. We further define a directed version $\vec{G}$ of the graph $G$, where every edge $e\notin M$ is directed from set $A$ to set $B$ and every edge $e\in M$ is directed from $B$ to $A$ (note that by definition of $A$ and $B$, every edge of $G$ is between a node in $A$ and a node in $B$). We then apply the following algorithm to compute a set $S$, which we will show is a $(1-1/k)$-approximate vertex cover of $G$.

**Basic Approximate Vertex Cover Algorithm**

1. Let $A_0\subseteq A$ be the set of unmatched nodes in $A$.
2. For every $i\in\{1,\dots,k\}$, let $A_i\subseteq A$ be the set of nodes in $A$ for which the shortest directed path in $\vec{G}$ from a node in $A_0$ is of length $2i$.
3. For every $i\in\{1,\dots,k\}$, let $B_i\subseteq B$ be the set of nodes in $\vec{G}$ from a nodes in $A_0$ is of length $2i-1$.
4. Define $i^*:=\arg\min_{i\in\{1,\dots,k\}}|B_i|$.
5. Output $S:=\bigcup_{i=1}^{i^*}B_i\cup\left(A\setminus\bigcup_{i=0}^{i^*-1}A_i\right)$.

▶ **Lemma 4.** *If the given matching $M$ has no augmenting paths of length at most $2k-1$, the above algorithm computes a vertex cover $S$ of $G$ of size at most $(1+1/k)\cdot\mathsf{OPT}$, where $\mathsf{OPT}$ is the size of a minimum vertex cover of $G$.*

**Proof.** We first show that $S$ is a vertex cover of $G$. A bit more generally, for any $\hat{\imath}\in\{1,\dots,k\}$, we define $S_{\hat{\imath}}:=\bigcup_{i=1}^{\hat{\imath}}B_i\cup\left(A\setminus\bigcup_{i=0}^{\hat{\imath}-1}A_i\right)$ and show that $S_{\hat{\imath}}$ is a vertex cover of $G$. For $S_{\hat{\imath}}$ to not be a vertex cover, there must be an a node $u$ in a set $A_j$ for $j\in\left\{0,\dots,\hat{\imath}-1\right\}$ and a node $v$ in $B\setminus\bigcup_{j=1}^{\hat{\imath}}B_j$. Note that the edge $\{u,v\}$ cannot be a matching edge because either $u\in A_0$, in which case $u$ is unmatched, or $u\in A_j$ for $j<\hat{\imath}$. In the second case, $u$ is reached over a path of length $2j$ in the directed graph $\vec{G}$ and thus $u$'s matching edge connects to a node in $B_j$. However, if $\{u,v\}$ is not a matching edge, it means that the edge $\{u,v\}$ is directed from $u$ to $v$ in graph $\vec{G}$. Therefore, since the shortest directed path from $A_0$ to $u$ in $\vec{G}$ is of length $2j$, there must be a directed path from $A_0$ to $v$ of length at most $2j+1$ in $\vec{G}$. This means that $v$ must be in one of the sets $B_1,\dots,B_{j+1}$. Since $j<\hat{\imath}$, this contradicts the assumption that $v\in B\setminus\bigcup_{j=1}^{\hat{\imath}}B_j$. We can therefore conclude that $S_{\hat{\imath}}$ is a vertex cover of $G$ for every $\hat{\imath}\in\{1,\dots,k\}$ and thus, in particular, the set $S_{i^*}$ is a vertex cover of $G$.

It remains to show that the size of $S$ is at most $(1+1/k)\cdot\mathsf{OPT}$. To prove this, we first show that for all $i\in\{1,\dots,k\}$, the set of nodes in $B_i$ are matched nodes. If there is a node $v\in B_i$ for $i\leq k$ that is unmatched, there is a directed path of length $2i-1\leq 2k-1$ in $\vec{G}$ from a node in $A_0$ to $v$. Such a directed path would be an augmenting path of the same length $2i-1\leq 2k-1$ w.r.t. matching $M$ in $G$. This cannot be because we assumed that there are no augmenting paths of length at most $2k-1$ w.r.t. $M$ in $G$. Further, by definition of the directed graph $\vec{G}$, the reason that a node $u$ is in a set $A_i$ for $i \in \{1, \dots, k\}$ is that the matching edge of $u$ connects $u$ to a node in $B_i$. By induction on $i$, we can therefore conclude that for all $i \in \{1, \dots, k\}$, the nodes in $A_i$ are exactly the matching neighbors of the nodes in $B_i$ and therefore for all such $i$, we have $|A_i| = |B_i|$. For every $\hat{i} \in \{1, \dots, k\}$, the size of the set $S_{\hat{i}}$ can therefore be computed as

$$|S_{\hat{i}}| = \sum_{i=1}^{\hat{i}} |B_i| + \underbrace{|A| - |A_0|}_{=|M|} - \sum_{i=1}^{\hat{i}-1} |A_i| = |M| + |B_{\hat{i}}|.$$

Because the sets $B_i$ for $i \in \{1, \dots, k\}$ are disjoint and they all contain matched nodes, their total size is at most $|M|$ and therefore, we have $|B_{i^*}| \leq |M|/k$. We can therefore conclude that $|S| = |M| + |B_{i^*}| \leq (1 + 1/k) \cdot |M| = (1 + 1/k) \cdot \mathsf{OPT}$. ◀

We next discuss how the above algorithm can efficiently be implemented in time $O(D + \text{poly}\log n)$ in the CONGEST model, where $D$ is the diameter of the graph. A bit more precisely, we will show the following. Let $G = (V, E)$ be a bipartite graph with diameter $D$ and let $G' = (V', E')$ be a subgraph of $G$. Assume that each node of $G$ knows if it is contained in the set $V'$ and which of its edges are contained in the set $E'$. We then show that for any $k \geq 1$, one can run the above algorithm on graph $G'$ in $O(D + k)$ rounds in the CONGEST model on graph $G$. The implementation is relatively straightforward. In time $O(D)$, one can compute a BFS tree of the graph $G$, and one can compute the bipartition of the nodes into sets $A$ and $B$. Then, in $O(k)$ rounds, one can do the BFS traversal on the directed graph $\vec{G}$, starting from nodes in $A_0$ and computing the sets $A_i$ and $B_i$ for $i \in \{1, \dots, k\}$. Finally, by using the BFS tree on graph $G$ and a simple pipelining scheme, one can compute the sizes of all the sets $B_i$ and determine the index $i^*$ of the smallest such set. A formal statement is given by the following lemma and a formal proof of the lemma can be found in the full version of the paper [13].

▶ **Lemma 5.** *Let $G = (V, E)$ be a bipartite graph of diameter $D$, let $G' = (V', E')$ be a subgraph of $G$ (i.e., $V' \subseteq V$ and $E' \subseteq E$), and let $k \geq 1$ be an integer parameter. Assume that $M$ is a matching of $G'$ s.t. there exists no augmenting path of length at most $2k - 1$ w.r.t. $M$ in $G'$. Then, there exists a deterministic* CONGEST *model algorithm to compute a $(1 + 1/k)$-approximate minimum vertex cover of $G'$ in $O(D + k)$ rounds on graph $G$.*

In combination with a distributed approximate maximum matching algorithm of Lotker, Patt-Shamir, and Pettie [28], Lemma 5 directly leads to a randomized $O(D + \text{poly}\log n)$-round distributed approximation scheme for the MVC problem.

▶ **Theorem 6.** *Let $G = (V, E)$ be a bipartite graph of diameter $D$ and $G' = (V', E')$ be a subgraph of $G$ (i.e., $V' \subseteq V$ and $E' \subseteq E$). For $\varepsilon \in (0, 1]$, there is a randomized algorithm that gives a $(1 + \varepsilon)$-approximate minimum vertex cover of $G'$ w.h.p. in $O(D + \frac{\log n}{\varepsilon^3})$ rounds in the* CONGEST *model on $G$.*

**Proof.** The approximate maximum matching algorithm of [28] is based on the classic approach of Hopcroft and Karp [21]. For a given graph and positive integer parameter $k$, the algorithm computes a matching $M$ of the graph such that there is no augmenting path of length at most $2k - 1$ w.r.t. $M$. When run on an $n$-node graph, the algorithm w.h.p. has a time complexity of $O(k^3 \cdot \log n)$ in the CONGEST model. The theorem therefore directly follows by applying the algorithm of [28] on $G'$ with $k = \lceil 1/\varepsilon \rceil$ and by Lemma 5. ◀

## 3.1 Deterministic MVC Approximation

The only part in the algorithm underlying Theorem 6 that is randomized is the approximate maximum matching algorithm of [28]. In order to also obtain a deterministic distributed MVC algorithm, we therefore have to replace the randomized distributed matching algorithm by a deterministic distributed matching algorithm. The algorithm of [28] is based on the framework of [21] and it therefore guarantees that the resulting matching has no short augmenting paths. While the size of such a matching is guaranteed to be close to the size of a maximum matching, the converse is not necessarily true.[1] Unfortunately, we are not aware of an efficient deterministic CONGEST model algorithm to compute a matching $M$ with no short augmenting paths. To resolve this issue, we therefore have to do some additional work.

For $\varepsilon > 0$, we define an augmenting path w.r.t. a matching in $G'$ to be short if it is of length at most $\ell = 2k' - 1$, where $k' = \lceil 2/\varepsilon \rceil$. We define $\delta \leq \varepsilon/(2\alpha)$ where $\alpha = O\big(\frac{\log \Delta}{\varepsilon^3}\big)$. We first run a polylogarithmic-time deterministic CONGEST algorithm by Ahmadi et al. [1] to obtain a $(1-\delta)$-approximate maximum matching $M$ in $G'$. This matching $M$ can potentially have short augmenting paths. In order to get rid of short augmenting paths, we then find a subset of nodes $S_1$ such that after deleting the nodes in $S_1$, $M$ is a matching with no short augmenting paths in the remaining subgraph $G''$ of $G'$. We show that we can select $S_1$ such that $|S_1| \leq \alpha\delta\mathsf{OPT}$, where $\mathsf{OPT}$ is the size of a minimum vertex cover in $G'$. Now that we end up with a matching in $G''$ with no short augmenting paths, we can directly apply our subroutine from above on $G''$ and obtain a set $S_2$ which is a $(1 + \frac{\varepsilon}{2})$-approximate vertex cover of $G''$. Finally, we deduce that $C = S_1 \cup S_2$ is a vertex cover of $G'$. Moreover, since the size of the minimum vertex cover of $G''$ is at most $\mathsf{OPT}$, we get $|C| = |S_1| + |S_2| \leq \alpha\delta\mathsf{OPT} + (1 + \frac{\varepsilon}{2})\mathsf{OPT} = (1 + \varepsilon)\mathsf{OPT}$.

**Finding $S_1$.** We next describe an algorithm to compute the set $S_1$. We assume that we are given an arbitrary $(1-\delta)$-approximate matching $M$ of $G' = (U' \cup V', E')$. As discussed above, we need to find a node set $S_1 \subseteq U' \cup V'$ that allows to get rid of augmenting paths of length at most $\ell = 2k' - 1$. This will be done in $(\ell + 1)/2$ stages $d = 1, 3, \ldots, \ell$. The objective of stage $d$ is to get rid of augmenting paths of length exactly $d$. Note that this guarantees that when starting stage $d$, there are no augmenting paths of length less than $d$ and thus in stage $d$, all augmenting paths of length $d$ are also shortest augmenting paths. In the following, we focus on a single stage $d$. Formally, the subproblem that we need to solve in stage $d$ is the following.

We are given a bipartite graph $H = (U_H \cup V_H, E_H)$ with at most $n$ nodes and we are given a matching $M_H$ of $H$. We assume that the bipartition of the graph into $U_H$ and $V_H$ is given. Let $d$ be a positive odd integer and assume that $H$ has no augmenting paths of length shorter than $d$ w.r.t. $M_H$. The goal is to find a set $S_H \subseteq U_H \cup V_H$ that is as small as possible such that when removing the set $S_H$ from the nodes of $H$ and the resulting induced subgraph $H' := H[U_H \cup V_H \setminus S_H]$ has no augmenting paths of length at most $d$ w.r.t. the matching $M'_H := M_H \cap E(H')$, i.e., w.r.t. to the matching induced by $M_H$ in the induced subgraph $H'$ of the remaining nodes.

We therefore need to find a set $S_H$ of nodes of $H$ such that $S_H$ contains at least one node of every augmenting path of length $d$ w.r.t. $M_H$ in graph $H$. Further, we want to make sure that after removing $S_H$, in the remaining induced subgraph $H'$ w.r.t. the remaining matching

---

[1] One can for example obtain an almost-maximum matching $M$ for some graph $G$ by taking a maximum matching of $G$ and flipping an arbitrary matched edge to unmatched. While the matching $M$ is obviously a very good approximate matching, it has a short augmenting path of length 1.

$M'_H$, there are no augmenting paths that were not present in graph $H$ w.r.t. matching $M_H$. To guarantee this, we make sure that whenever we add a matched node in $U_H \cup V_H$ to $S_H$, we also add its matched neighbor to $S_H$. In this way, every node that is unmatched in $H'$ was also unmatched in $H$ and therefore any augmenting path in $H'$ is also an augmenting path in $H$.

**Getting Rid of Short Augmenting Paths by Solving Set Cover.** The problem of finding a minimal such collection of matching edges and unmatched nodes can be phrased as a minimum set cover problem. The ground set $\mathcal{P}$ is the set of all augmenting paths of length $d$ w.r.t. $M_H$ in $H$. For each unmatched node $v \in U_H \cup V_H$, we define $P_v$ as the set of augmenting paths of length $d$ that contain $v$. Similarly, for each matching edge $e \in M_H$, we define $P_e$ as the set of augmenting paths of length $d$ that contain $e$. The goal is to find a smallest set $C$ consisting of unmatched nodes $v$ in $U_H \cup V_H$ and matching edges $e \in M_H$ such that the union of the corresponding sets $P_v$ and $P_e$ of paths covers all paths in $\mathcal{P}$. The set $S_H$ then consists of all nodes in $C$ and both nodes of each edge in $C$. Let us first have a look at the structure of augmenting paths of length $d$ in $H$. Let $L_0$ be the set of unmatched nodes in $U_H$ and more generally let $L_i \subseteq U_H \cup V_H$ for $i \in \{0, \dots, d\}$ be the set of nodes of $H$ that can be reached over a shortest alternating path of length $i$ from a node in $L_0$. Since the bipartition into $U_H$ and $V_H$ is given, the sets $L_0, \dots, L_d$ can be computed in $d$ CONGEST rounds by a simple parallel BFS exploration. Since we assume that $H$ has no augmenting paths of length shorter than $d$, every augmenting path of length $d$ contains exactly one node from every set $L_i$ such that the node in $L_d$ is an unmatched node in $V_H$.

We use a variant of the greedy set cover algorithm to find the set $C$ covering all the shortest augmenting paths in $H$. In order to apply the greedy set cover algorithm, we need to know the sizes of the sets $P_v$, i.e., for every node $v$, we need to know in how many augmenting paths of length $d$ the node $v$ is contained. To compute this number, we apply an algorithm that was first developed in [28] and later refined in [5]. The following lemma summarizes the result of [5, 28], for a proof see also the full version of this paper [13].

▶ **Lemma 7.** *[5, 28] Let $H = (U_H \cup V_H, E_H)$ be a bipartite graph of maximum degree at most $\Delta$ and $M_H$ be a matching of $H$. There is a deterministic $O(d^2)$-round* CONGEST *algorithm to compute the number of shortest augmenting paths of length $d$ passing through every node $v \in U_H \cup V_H$.*

We can now use this path counting method to find a small set $S$ of nodes that covers all augmenting paths of length $d$. We start with an empty set $C$. The algorithm then works in $O(d \log \Delta)$ phases $i = 1, 2, 3, \dots$, where in phase $i$, we add unmatched nodes $v$ and matching edges $e$ to $C$ such that are still contained in at least $\Delta^d/2^i$ remaining paths. In order to obtain a polylogarithmic running time, we need to add nodes and edges to $C$ in parallel. In order to make sure that we do not cover the same path twice, when adding nodes and edges in parallel, we essentially iterate through the $d$ levels in each phase. The details of the algorithm are given in the following.

**Covering Paths of Length $d$: Phase $i \geq 1$**

Iterate over all odd levels $\ell = 1, 3, \ldots, d$:

1. Count the number of augmenting paths of length $d$ passing through each of the remaining nodes and edges.
2. If $\ell \in \{1, d\}$, for all remaining nodes $v \in L_\ell$ that are in $p_v \geq \Delta^d/2^i$ different augmenting paths of length $d$, add $v$ to $C$ and remove $v$ and its incident edges from $G_H$ for the remainder of the algorithm.
3. If $\ell \in \{2, \ldots, d-1\}$, for all remaining matching edges $e \in M_H$ connecting two nodes $u \in L_{\ell-1}$ and $v \in L_\ell$ that are in $p_e \geq \Delta^d/2^i$ different augmenting paths of length $d$, add $e$ to $C$ and remove $e$ and its incident edges from $G_H$ for the remainder of the algorithm.

Define $S_H$ to contain every node in $C$ and both nodes of every edge in $C$.

▶ **Lemma 8.** *Let $\delta \in (0,1)$ and assume that $M_H$ is a $(1-\delta)$-approximate matching of the bipartite graph $H$ of maximum degree at most $\Delta$ . Then, the set $S_H$ selected by the above algorithm has size at most $\alpha_d \delta \cdot \mathsf{OPT}_H$, where $\alpha_d = 2(d+3)(1 + d \ln \Delta)$ and $\mathsf{OPT}_H$ is the size of a maximum matching and thus of a minimum vertex cover of $H$. The time complexity of the algorithm in the* CONGEST *model is $O(d^4 \log \Delta)$.*

**Proof.** We first look at the time complexity of the algorithm in the CONGEST model. The algorithm consists of $O(d \log \Delta)$ phases, in each phase, we iterate over $O(d)$ levels and in each of these iterations, the most expensive step is to count the number of augmenting paths passing through each node and edge. By Lemma 7, this can be done in time $O(d^2)$, resulting in an overall time complexity of $O(d^4 \log \Delta)$.

For each free node $v \in U_H \cup V_H$ and for each matching edge $e \in M_H$, let $p_v$ and $p_e$ be the number of (uncovered) augmenting paths of length $d$ passing through $v$ and $e$, respectively. We will next show that our algorithm is simulating a version of the standard sequential greedy set cover algorithm. When applying the sequential greedy algorithm, in each step, we would need to choose a set $P_v$ or $P_e$ of paths that maximizes the number of uncovered augmenting paths of length $d$ the set covers. We will see that we essentially relax the greedy step and we obtain an algorithm that is equivalent to a sequential algorithm that always picks a set of paths that contains at least half as many uncovered paths as possible. To show this, we first show that for each phase $i$, at the beginning of the phase, we have $p_v, p_e \leq \Delta^d/2^{i-1}$ for all unmatched nodes $v$ and matching edges $e$. For the sake of contradiction, assume that this is not the case and let $i'$ be the first phase, in which it is not true. Because every node and edge can be contained in at most $\Delta^d$ augmenting paths of length $d$, the statement is definitely true for the first phase and we therefore have $i' > 1$. We now consider phase $i' - 1$. In each phase, by iterating over all odd levels $\ell = 1, 3, \ldots, d$, we iterate over all unmatched nodes $v \in U_H \cup V_H$ and all matching edges $e \in M_H$ that are contained in some augmenting path of length $d$. For each of them, we add the corresponding set $P_v$ or $P_e$ to the set cover if we still have $p_v \geq \Delta/2^{i'-1}$ or $p_e \geq \Delta/2^{i'-1}$. At the end of phase $i' - 1$, we therefore definitely have $p_v, p_e < \Delta/2^{i'-1}$ for all nodes $v$ and matching edges $e$, which contradicts the assumption that at the beginning of phase $i'$, it is not true that $p_v, p_e \leq \Delta/2^{i'-1}$ for all such $v$ and $e$. Because in each phase $i$, we only add set $P_v$ and $P_e$ that are contained in at least $\Delta/2^i$ uncovered paths, we clearly always pick sets that cover at least half as many uncovered paths as the best current set. Note also that because we iterate through the levels and only add sets for nodes or edges on the same level in parallel, the set that we add in parallel cover disjoint sets of paths. The algorithm is therefore equivalent to a sequential algorithm that adds the sets in each parallel step in an arbitrary order.

Now, we will show that we remove at most $2(d+3)(1+d\ln\Delta)\delta\cdot\mathsf{OPT}_H$ nodes from graph $H$. Indeed, approximating the set cover problem using the standard greedy algorithm gives a $(1+\ln(s))$ approximation to the solution, where $s$ is the cardinality of the largest set. If we relax the greedy step by at least a factor of two, as our algorithm does, a standard analysis implies that we still get a $2(1+\ln s)$-approximation of the corresponding minimum set cover problem, where $s$ is still defined as the cardinality of the largest set. In our case, the largest set $P_v$ or $P_e$ is $s\leq\Delta^d$. Now if the solution to the set cover problem using this greedy version algorithm is $S_H$ and the optimal solution of the set cover problem is $S^*$, then $|S^*|\leq|S_H|\leq 2(1+d\ln\Delta)|S^*|$. Recall that $P_e$ corresponds to a matched edge and by step 3 in our algorithm, both of these matched nodes are removed from the graph $H$. Hence, we remove up to $2|S_H|\leq 4(1+d\ln\Delta)|S^*|$ nodes from $H$.

Next, we give an upper bound to $|S^*|$, which will finish up our proof. Recall that a solution to our set cover problem is a set of matched edges $S_e$ and a set of unmatched nodes $S_v$ that cover all augmenting paths of length $d$ in $H$, i.e., all paths in $\mathcal{P}$. Luckily, there is a simple solution to the given set cover problem that allows us to upper bound $|S^*|$. We just select a maximal set $P$ of vertex-disjoint augmenting paths of length $d$ and we consider all the unmatched nodes and matched edges on these paths to be our solution $S'$, where $|S'|=\frac{d+3}{2}|P|$. Clearly, $S'$ is a set cover (and thus $|S^*|\leq|S'|$), as otherwise there would be an augmenting path of length $d$ that is not covered by $S'$. This path has to be vertex-disjoint from all the paths in $P$, which is a contradiction to the assumption that $P$ is a maximal set of vertex-disjoint augmenting paths of length $d$. Let $|M_H^*|$ denote the maximum cardinality of a matching of graph $H$. Now, since $M_H$ is a $(1-\delta)$-approximate matching, we can clearly have at most $\delta|M_H^*|$ vertex-disjoint augmenting paths of at most length $d$. Hence, the size of $P$ can never exceed $\delta|M_H^*|$ i.e. $|P|\leq\delta|M_H^*|$. Thus, $|S^*|\leq|S'|\leq\frac{d+3}{2}\delta|M_H^*|$. Hence, we remove at most $2|S_H|\leq 4(1+d\ln\Delta)|S'|\leq 4(1+d\ln\Delta)\frac{d+3}{2}\delta|M_H^*|\leq 2(d+3)(1+d\ln\Delta)\delta|M_H^*|=2(d+3)(1+d\ln\Delta)\delta\cdot\mathsf{OPT}_H$ nodes from graph $H$. ◀

By iterating over the lengths of shortest paths, we now directly get the following lemma. For a formal proof of the lemma, we refer to the full version of this paper [13].

▶ **Lemma 9.** *Let $G=(U\cup V,E)$ be a bipartite graph, let $k\geq 1$ be an integer parameter, and assume that $M$ is a $(1-\delta)$-approximate matching of $G$ for some $\delta\in[0,1]$. Further, let $\mathsf{OPT}$ be the size of a minimum vertex cover of $G$. If the bipartition of the nodes of $G$ into $U$ and $V$ is given, there is an $O(k^5\log\Delta)$-time algorithm to compute a node set $S_1\subseteq U\cup V$ of size at most $4k(k+1)(1+2k\ln\Delta)\delta\cdot\mathsf{OPT}$ such that in the induced subgraph $G[U\cup V\setminus S]$, there is no augmenting path of length at most $2k-1$ w.r.t. the matching $\bar{M}$, where $\bar{M}\subseteq M$ consists of the edges of $M$ that connect two nodes in $U\cup V\setminus S_1$.*

We now have everything that we need to also get a deterministic $O\big(D+\text{poly}\left(\frac{\log n}{\varepsilon}\right)\big)$-time $\mathsf{CONGEST}$ algorithm for computing a $(1+\varepsilon)$-approximate solution for the MVC problem in bipartite graphs.

▶ **Theorem 10.** *Let $G=(V,E)$ be a bipartite graph of diameter $D$ and maximum degree $\Delta$ and let $G'=(V',E')$ be a subgraph of $G$. For $\varepsilon\in(0,1]$, there is a deterministic algorithm that gives a $(1+\varepsilon)$-approximate minimum vertex cover of graph $G'$ in $O(D+\frac{log^4 n}{\varepsilon^8})$ rounds in the $\mathsf{CONGEST}$ model on $G$.*

**Proof.** As a first step, we choose a sufficiently small parameter $\delta>0$ and we compute a $(1-\delta)$-approximate solution $M'$ to the maximum matching problem on $G'$ by using the deterministic $\mathsf{CONGEST}$ algorithm of [1]. For computing such a matching, the algorithm of [1] has a time complexity of $O\big(\frac{\log^2\Delta+\log^* n}{\delta}+\frac{\log\Delta}{\delta^2}\big)=O\big(\frac{\log^2 n}{\delta^2}\big)$. Let $k':=\lceil 2/\varepsilon\rceil$ as discussed above. By

Lemma 9, there is a value $\alpha = 4k'(k'+1)(1+2k'\ln\Delta) = O(k'^3 \log\Delta)$ such that we can find a set $S_1 \subseteq V'$ of size $|S_1| = \alpha\delta\mathsf{OPT}$, where $\mathsf{OPT}$ is the size of a minimum vertex cover of $G'$, such that the following is true. The set $S_1$ can be computed in time $O(k'^5 \log\Delta) = O\big(\frac{\log n}{\varepsilon^5}\big)$. Let $G'' = G'[V' \setminus S_1]$ be the induced subgraph of $G'$ after removing all the nodes in $S_1$ and let $M''$ be the subset of the edges in $M'$ that connect two nodes in $V' \setminus S_1$ (i.e., $M''$ is a matching of $G''$). Then, the graph $G''$ has no augmenting paths of length at most $2k'-1$. By using Lemma 5, we can therefore compute a $(1+1/k')$-approximate vertex cover $S_2$ (and thus a $(1+\varepsilon/2)$-approximate vertex cover) of $G''$ in time $O(D+k') = O(D+1/\varepsilon)$. Because a minimum vertex cover of $G''$ is clearly not larger than a minimum vertex cover of $G'$, we therefore have $|S_2| \leq (1+\varepsilon/2)\cdot\mathsf{OPT}$. Note that $S_1 \cup S_2$ is a vertex cover of $G'$. The size of $S_1 \cup S_2$ can be bounded as $|S_1 \cup S_2| \leq \delta\alpha\cdot\mathsf{OPT} + (1+\varepsilon/2)\cdot\mathsf{OPT}$. In order to make sure that this is at most $(1+\varepsilon)\cdot\mathsf{OPT}$, we have to choose $\delta \leq \varepsilon/(2\alpha)$. The time complexity to compute the initial matching $M'$ of $G'$ is therefore $O\big(\frac{\log^2 n}{\delta^2}\big) = O\big(\frac{\log^4 n}{\varepsilon^8}\big)$. ◀

## 4 Polylogarithmic-Time Algorithms

We next show how we can use the algorithms of the previous section together with existing low-diameter graph clustering techniques to obtain polylogarithmic-time approximation schemes for the minimum vertex cover algorithm in the CONGEST model. First we describe a general framework for achieving a $(1+\varepsilon)$-approximate minimum vertex cover $C$ of unweighted bipartite graphs via an efficient algorithm in the CONGEST model based on a given clustering with some specific properties (cf. Section 2 for the corresponding definitions). We will do so by proving the following lemma. Note that our general framework applies to both the randomized and the deterministic case.

▶ **Lemma 11.** *Let $G=(V,E)$ be a bipartite graph and assume that we are given a maximal matching $M$ of $G$. We define edge weights $w(e) \in \{0,1\}$ such that $w(e)=1$ if and only if $e \in M$. Further, assume that w.r.t. those edge weights, we are given a $(1-\eta)$ dense, 3-hop separated, and $(c,d)$-routable clustering of $G$, for some $\eta \in (0,1]$ and some positive integers $c,d>0$. Then, for any $\psi \in (0,1]$, we can find a $(1+2\eta+\psi)$-approximate minimum vertex cover by a deterministic* CONGEST *algorithm in $O\big(c\cdot\big(d+\operatorname{poly}\frac{\log n}{\psi}\big)\big)$ rounds and by a randomized* CONGEST *algorithm in $O\big(c\cdot\big(d+\frac{\log n}{\psi^3}\big)\big)$ rounds, w.h.p.*

**Proof.** Let $\{S_1, S_2, ..., S_t\}$ be the collection of clusters of the given 3-hop separated, $(1-\eta)$-dense clustering. Define $E'$ to be the set of edges for which both endpoints are located outside clusters and let $E''$ to be the set of edges where exactly one of the endpoints is outside clusters. We also say that $e$ is an edge outside clusters if it is in $E' \cup E''$. Further, let $X$ to be the set of all matched nodes (w.r.t. the given maximal matching $M$) that are outside clusters. Note that since $M$ is a maximal matching, any edge in $E'$ is necessarily incident to at least one matched node of $M$. Therefore, when adding the set $X$ to the vertex cover $C$, we cover all edges in $E'$ and possibly some extra edges in $E''$. Now since $G$ is $(1-\eta)$-dense, then at most $\eta|M|$ matched edges are outside clusters, and when assuming that $|M^*|$ is the size of a maximum matching of $G$, we can deduce that $|X| \leq 2\eta|M| \leq 2\eta|M^*| = 2\eta\mathsf{OPT}$, where $\mathsf{OPT}$ is the size of a minimum vertex cover of $G$. Next, we extend each cluster $S_i$ by at most one hop in radius as follows. For every edge $\{u,v\} \in E''$ such that $u \in S_i$ and $v \notin S_i$, we add the edge $\{u,v\}$ and node $v$ to the cluster. Let $\{S'_1, S'_2, ..., S'_t\}$ be the new collection of extended clusters. All edges of $G$ that are not already covered by $X$ are now inside some cluster. In addition, we grow the height of each cluster tree $T_i$ by at most one hop so that they include the new cluster nodes. We denote the new extended trees by $T_i'$. Note that clearly, each edge in $E$ is still in at most $c$ trees. Hence, the new collection of extended clusters are now 2-hop separated and $(c, d+1)$-routable.

For each cluster $S_i'$, let $G_i'$ be the graph consisting of the nodes and edges of the cluster. We note that because the clusters are 1-hop separated, the graphs $G_i'$ are vertex and edge disjoint. In addition, for each cluster $S_i'$, we define the graph $G_i$ as the union of $G_i'$ and the tree $T_i'$. Because the clustering is $(c, d+1)$-routable, it follows that every edge of $G$ is used by at most $c$ of the graph $G_i$ and that the diameter of each graph $G_i$ is at most $d+1$. To obtain a vertex cover of all edges of $G$, we now compute a $(1+\psi)$-approximate minimum vertex cover $C_i$ for each extended cluster graph $G_i'$ by running the algorithms described in Theorems 6 and 10. We do this for all clusters in parallel. For each cluster $S_i'$, we use $G_i$ and $G_i'$ as the graphs $G$ and $G'$ in Theorems 6 and 10. Because each edge is contained in at most $c$ graphs $G_i$, we can in parallel run $T$-round algorithms in all graphs $G_i$ in time $c \cdot T$. The time complexities therefore follow directly as claimed from the respective time complexities in Theorems 6 and 10.

We define $Y := \bigcup_{i=1}^{t} C_i$. Because every edge of $G$ that is not covered by the nodes in $X$ is inside one of the clusters $S_i'$, clearly, the set $X \cup Y$ is a vertex cover of $G$. We already showed that $|X| \leq 2\eta\mathsf{OPT}$. To bound the size of $X \cup Y$, it remains to bound the size of $Y$. Let $\mathsf{OPT}_i$ be the size of an optimal vertex cover of $G_i'$. Because the cluster graphs $G_i'$ are vertex-disjoint, all edges in $G_i'$ clearly have to be covered by some node of the cluster $S_i'$ and thus edges in different clusters have to be covered by disjoint sets of nodes. If $\mathsf{OPT}$ is the size of an optimal vertex cover of $G$, we thus clearly have $\bigcup_{i=1}^{t} \mathsf{OPT}_i \leq \mathsf{OPT}$. Because $C_i$ is a $(1+\psi)$-approximate vertex cover of $G_i'$, we also have $|C_i| \leq (1+\psi) \cdot \mathsf{OPT}_i$. Together, we therefore directly get that $|Y| \leq (1+\psi) \cdot \mathsf{OPT}$ and therefore $|X \cup Y| \leq (1+2\eta+\psi) \cdot \mathsf{OPT}$. ◀

In order to prove our two main results, Theorems 2 and 3, we will next show how to efficiently compute the clusterings that are required for Lemma 11. Both clusterings can be obtained by minor adaptations of existing clustering techniques.

## 4.1 The Randomized Clustering

We start with describing the randomized clustering algorithm. By using the exponentially shifted shortest paths approach of Miller, Peng, and Xu [30], we obtain the following lemma.

▶ **Lemma 12.** *Let $G = (V, E, w)$ be a weighted bipartite graph with non-negative edge weights $w(e)$. For $\lambda \in (0, 1]$, there is a randomized algorithm that computes a 3-hop separated clustering of $G$ such that w.h.p., the clustering is $(1, O(\frac{\log n}{\lambda}))$-routable and can be computed in $O(\frac{\log n}{\lambda})$ rounds in the $\mathsf{CONGEST}$ model and such that the clustering is $(1-\lambda)$-dense in expectation.*

The proof of Lemma 12 is a relatively simple adaptation of the clustering algorithm of [30]. For a proof, see the full version of this paper [13].

We now have everything that we need to prove our first main result, our randomized polylogarithmic-time approximation scheme for the MVC problem in bipartite graphs.

**Proof of Theorem 2.** Let $G = (V, E)$ be the given bipartite graph for which we want to approximate the MVC problem. We first compute a maximal matching $M$ of $G$, which we can for example do by using Luby's algorithm [2, 29] in $O(\log n)$ rounds. By using $M$, we then apply Lemma 12 with $\lambda = \varepsilon/4$ to obtain a 3-hop separated $\big(1, O\big(\frac{\log n}{\varepsilon}\big)\big)$-routable clustering that is $(1-\varepsilon/4)$-dense in expectation. The time for computing the clustering is $O\big(\frac{\log n}{\varepsilon}\big)$, w.h.p. By applying Lemma 11 with $\eta = \varepsilon/4$ and $\psi = \varepsilon/2$, we then get a vertex cover of $G$ in $O\big(\frac{\log n}{\varepsilon^3}\big)$ CONGEST rounds such that the expected size of the vertex cover is at most $(1+\varepsilon)\cdot \mathsf{OPT}$, where $\mathsf{OPT}$ is the size of a minimum vertex cover of $G$. This concludes the proof of the theorem. ◀

## 4.2 The Deterministic Clustering

We obtain the deterministic version of the necessary clustering by adapting the construction of a single color class of the recent efficient deterministic network decomposition algorithm of Rozhoň and Ghaffari [32].

▶ **Lemma 13.** *Let $G = (V, E, w)$ be a weighted bipartite graph with non-negative edge weights $w(e) \in \{0, 1\}$. For $\lambda \in (0, 1]$, there is a deterministic algorithm that computes an $(1-\lambda)$-dense, 3-hop separated, and $\big(O(\log n), O\big(\frac{\log^3 n}{\lambda}\big)\big)$-routable clustering of $G$ in $\mathrm{poly}\big(\frac{\log n}{\lambda}\big)$ rounds in the* CONGEST *model.*

**Proof.** We assume that $W := \sum_{w\in E} w(e)$ is the total weight of all edges in $G$. Let $\lambda \in (0, 1]$. We adapt the weak diameter network decomposition algorithm of Rozhoň and Ghaffari [32] applied to the graph $G^2$ in the CONGEST model. When applied to $G^2$, Theorem 2.12 of [32] shows that the algorithm of [32] computes a decomposition of the nodes $V$ into clusters of $O(\log n)$ colors such that any two nodes in different clusters of the same color are at distance at least 3 from each other (in $G$). Each cluster is spanned by a Steiner tree of diameter $O(\log^3 n)$ such that each edge of $G$ is used by at most $O(\log n)$ different Steiner trees for each of the $O(\log n)$ color classes. For our purpose, we only need to construct the first color class of this decomposition. For the first color class, the proof of Theorem 2.12 of [32] implies that the clusters of the first color are 3-hop separated and that they contain a constant fraction of all the nodes. We need to adapt the construction of the first color class of the algorithm of [32] in two ways. In the following, we only sketch these changes.

First, we adapt the algorithm so that it can handle weights. In the following, we define node weight $\nu(v) \geq 0$ as follows. For each node $v$, we define $\nu(v)$ as the sum of the weights $w(e)$ of the edges $e$ that are incident to $v$. Note that this implies that the total weight of all the nodes is $2W$ and that the total weight of all the nodes that are not clustered is an upper bound on the total weight of all the edges outside clusters (i.e., all the edges, where at most one endpoint is inside a cluster). In the algorithm of [32], the clustering is computed in different steps. In each step, some nodes request to join a different cluster and a cluster accepts these requests if the total number of nodes requesting to join the cluster is large enough compared to the total number of nodes already inside the cluster. If a cluster does not accept the requests, the requesting nodes are deactivated and will not be clustered. The threshold on the number of requests required to accept the requests is chosen such that in the end the weak diameter of the clusters is not too large and at the same time, only a constant fraction of all nodes are deactivated and thus not clustered. In our case, we do not care how many nodes are clustered and unclustered, but we care about the total weight of nodes that are clustered and unclustered. The analysis of [32] however directly also works if we instead compare the total weight of the nodes that request to join a cluster with the total weight of the nodes that are already inside the cluster. If the node weights are polynomially bounded non-negative integers (which they are in our case), the asymptotic guarantees of the construction are exactly the same. In this way, we can make sure to construct $(O(\log n), O(\log^3 n))$-routable, 3-hop separated clusters such that a constant fraction of the total weight of all the nodes is inside clusters.

As a second change, in order to make sure that the clustering is also $(1-\lambda)$-dense, we need to guarantee that the total weight of the nodes that are unclustered is at most a $\lambda/2$-fraction of the total weight of all the nodes. We can guarantee this, by adapting the threshold for accepting nodes to a cluster. We essentially have to multiply the threshold by a factor $\Theta(\lambda)$ to make sure that this is the case. This increases the maximal possible cluster diameter by a factor $O(1/\lambda)$ and it increases the total running time by a factor $\mathrm{poly}(1/\lambda)$. ◀

*Remark:* In the above lemma, we assumed for simplicity that the edge weights are either 0 or 1. The construction however directly also works in the same way and with the same asymptotic guarantees if the edge weights are polynomially bounded non-negative integers. With some simple preprocessing, one can also obtain the same asymptotic result for arbitrary non-negative edge weights.

In a similar way as we proved Theorem 2, we can now also prove our second main result, our deterministic polylogarithmic-time approximation scheme for the MVC problem in bipartite graphs.

**Proof of Theorem 3.** Let $G = (V, E)$ be the given bipartite graph for which we want to approximate the MVC problem. We first compute a maximal matching $M$ of $G$, which we can do by using the algorithm of Fischer [15] in $O(\log^2 \Delta \cdot \log n)$ deterministic rounds in the CONGEST model. By using $M$, we then apply Lemma 13 with $\lambda = \varepsilon/4$ to obtain a $(1-\varepsilon/4)$-dense, 3-hop separated $\big(O(\log n), \mathrm{poly}\big(\frac{\log n}{\varepsilon}\big)\big)$-routable clustering. By Lemma 13, the time for computing the clustering in the CONGEST model is $\mathrm{poly}\big(\frac{\log n}{\varepsilon}\big)$. By applying Lemma 11 with $\eta = \varepsilon/4$ and $\psi = \varepsilon/2$, we then get a $(1+\varepsilon)$-approximate vertex cover of $G$ in $\mathrm{poly}\big(\frac{\log n}{\varepsilon}\big)$ CONGEST rounds, which completes the proof of the theorem. ◀

## References


1 M. Ahmadi, F. Kuhn, and R. Oshman. Distributed approximate maximum matching in the CONGEST model. In *Proc. 32nd Symp. on Distributed Computing (DISC)*, pages 6:1–6:17, 2018.

2 N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986.

3 M. Åstrand, P. Floréen, V. Polishchuk, J. Rybicki, J. Suomela, and J. Uitto. A local 2-approximation algorithm for the vertex cover problem. In *Proc. 23rd Symp. on Distributed Computing (DISC)*, pages 191–205, 2009.

4 N. Bachrach, K. Censor-Hillel, M. Dory, Y. Efron, D. Leitersdorf, and A. Paz. Hardness of distributed optimization. In *Proc. 38th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 238–247, 2019.

5 R. Bar-Yehuda, K. Censor-Hillel, M. Ghaffari, and G. Schwartzman. Distributed approximation of maximum independent set and maximum matching. *CoRR*, abs/1708.00276, 2017. Conference version at PODC 2017.

6 R. Bar-Yehuda, K. Censor-Hillel, Y. Maus, S. Pai, and S. V. Pemmaraju. Distributed approximation on power graphs. In *Proc. 39th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 501–510, 2020.

7 R. Bar-Yehuda, K. Censor-Hillel, and G. Schwartzman. A distributed $(2+\varepsilon)$-approximation for vertex cover in o(log$\delta/\varepsilon$ log log $\delta$) rounds. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 3–8, 2016.

8 L. Barenboim, M. Elkin, S. Pettie, and J. Schneider. The locality of distributed symmetry breaking. In *Proceedings of 53th Symposium on Foundations of Computer Science (FOCS)*, 2012.

**9** R. Ben-Basat, G. Even, K. Kawarabayashi, and G. Schwartzman. Optimal distributed covering algorithms. In *Proc. 33rd Symp. on Distributed Computing (DISC)*, pages 5:1–5:15, 2019.

**10** G. E. Blelloch, A. Gupta, I. Koutis, G. L. Miller, R. Peng, and K. Tangwongsan. Nearly-linear work parallel SDD solvers, low-diameter decomposition, and low-stretch subgraphs. *Theory Comput. Syst.*, 55(3):521–554, 2014.

**11** K. Censor-Hillel, S. Khoury, and A. Paz. Quadratic and near-quadratic lower bounds for the CONGEST model. In *Proc. 31st Symp. on Distributed Computing (DISC)*, pages 10:1–10:16, 2017.

**12** R. Diestel. *Graph Theory*, chapter 2.1, pages 35–58. Springer, Berlin, 3rd edition, 2005.

**13** S. Faour and F. Kuhn. Approximate bipartite vertex cover in the CONGEST model. *CoRR*, abs/2011.10014, 2020.

**14** U. Feige, Y. Mansour, and R. E. Schapire. Learning and inference in the presence of corrupted inputs. In *Proc. 28th Conf. on Learning Theory (COLT)*, pages 637–657, 2015.

**15** M. Fischer. Improved deterministic distributed matching via rounding. In *Proc. 31st Symp. on Distributed Computing (DISC)*, pages 17:1–17:15, 2017.

**16** M. Ghaffari, C. Jin, and D. Nilis. A massively parallel algorithm for minimum weight vertex cover. In *Proc. 32nd ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 259–268, 2020.

**17** M. Ghaffari, F. Kuhn, and Y. Maus. On the complexity of local distributed graph problems. In *Proc. 39th ACM Symp. on Theory of Computing (STOC)*, pages 784–797, 2017.

**18** M. Göös and J. Suomela. No sublogarithmic-time approximation scheme for bipartite vertex cover. *Distributed Computing*, 27(6):435–443, 2014.

**19** F. Grandoni, J. Könemann, and A. Panconesi. Distributed weighted vertex cover via maximal matchings. *ACM Trans. Algorithms*, 5(1):6:1–6:12, 2008.

**20** F. Grandoni, J. Könemann, A. Panconesi, and M. Sozio. A primal-dual bicriteria distributed algorithm for capacitated vertex cover. *SIAM J. Comput.*, 38(3):825–840, 2008.

**21** J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 1973.

**22** A. Israeli and A. Itai. A fast and simple randomized parallel algorithm for maximal matching. *Inf. Process. Lett.*, 22(2):77–80, 1986.

**23** D. Kőnig. Gráfok és mátrixok. *Matematikai és Fizikai Lapok*, 38:116–119, 1931.

**24** Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.

**25** F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally! In *Proceedings of 23rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 300–309, 2004.

**26** F. Kuhn, T. Moscibroda, and R. Wattenhofer. The price of being near-sighted. In *Proceedings of 17th Symposium on Discrete Algorithms (SODA)*, pages 980–989, 2006.

**27** N. Linial and M. Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, 1993.

**28** Z. Lotker, B. Patt-Shamir, and S. Pettie. Improved distributed approximate matching. *J. ACM*, 62(5):38:1–38:17, 2015.

**29** M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15:1036–1053, 1986.

**30** G. L. Miller, R. Peng, and S. C. Xu. Parallel graph decompositions using random shifts. In *Proc. 25th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 196–203, 2013.

**31** D. Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM, 2000.

**32** V. Rozhoň and M. Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In *Proc. 52nd ACM Symp. on Theory of Computing (STOC)*, pages 350–363, 2020.