Distributed CONGEST Approximation of Weighted Vertex Covers and Matchings

Salwa Faour
University of Freiburg, Germany

Marc Fuchs
University of Freiburg, Germany

Fabian Kuhn
University of Freiburg, Germany

Abstract

We provide CONGEST model algorithms for approximating the minimum weighted vertex cover and the maximum weighted matching problem. For bipartite graphs, we show that a $(1+\varepsilon)$-approximate weighted vertex cover can be computed deterministically in poly $\left(\frac{\log n}{\varepsilon}\right)$ rounds. This generalizes a corresponding result for the unweighted vertex cover problem shown in [Faour, Kuhn; OPODIS '20]. Moreover, we show that in general weighted graph families that are closed under taking subgraphs and in which we can compute an independent set of weight at least $\lambda \cdot w(V)$ (where $w(V)$ denotes the total weight of all nodes) in polylogarithmic time in the CONGEST model, one can compute a $(2-2\lambda+\varepsilon)$-approximate weighted vertex cover in poly $\left(\frac{\log n}{\varepsilon}\right)$ rounds in the CONGEST model. Our result in particular implies that in graphs of arboricity a, one can compute a $(2-1/a+\varepsilon)$-approximate weighted vertex cover problem in poly $\left(\frac{\log n}{\varepsilon}\right)$ rounds in the CONGEST model.

For maximum weighted matchings, we show that a $(1-\varepsilon)$-approximate solution can be computed deterministically in time $2^{O(1/\varepsilon)} \cdot \operatorname{poly}\log n$ in the CONGEST model. We also provide a randomized algorithm that with arbitrarily good constant probability succeeds in computing a $(1-\varepsilon)$-approximate weighted matching in time $2^{O(1/\varepsilon)} \cdot \operatorname{poly}\log(\Delta W) \cdot \log^* n$, where W denotes the ratio between the largest and the smallest edge weight. Our algorithm generalizes results of [Lotker, Patt-Shamir, Pettie; SPAA '08] and [Bar-Yehuda, Hillel, Ghaffari, Schwartzman; PODC '17], who gave $2^{O(1/\varepsilon)} \cdot \log n$ and $2^{O(1/\varepsilon)} \cdot \frac{\log \Delta}{\log\log \Delta}$-round randomized approximations for the unweighted matching problem.

Finally, we show that even in the LOCAL model and in bipartite graphs of degree ≤ 3, if $\varepsilon < \varepsilon_0$ for some constant $\varepsilon_0 > 0$, then computing a $(1+\varepsilon)$-approximation for the unweighted minimum vertex cover problem requires $\Omega\left(\frac{\log n}{\varepsilon}\right)$ rounds. This generalizes a result of [Göös, Suomela; DISC '12], who showed that computing a $(1+\varepsilon_0)$-approximation in such graphs requires $\Omega(\log n)$ rounds.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases distributed graph algorithms, minimum weighted vertex cover, maximum weighted matching, distributed optimization, CONGEST model

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2021.17

Related Version *Full Version*: https://arxiv.org/abs/2111.10577

1 Introduction and Related Work

Maximum matching (MM) and minimum vertex cover (MVC) are two classic optimization problems that have been studied intensively in the context of distributed graph algorithms (e.g., [1–5, 7–11, 14, 15, 21, 23, 25, 28, 32–35, 38, 40, 45–47, 49, 50, 52, 56]). The problems are closely related to each other: the fractional relaxations of the unweighted variants of the problem are linear programming (LP) duals of each other. The problems are however also fundamentally different. While a maximum (weighted) matching can be found in polynomial time in all graphs [18, 19], for the minimum (weighted) vertex cover problem, this is only true for bipartite graphs [20, 42]. In general graphs, even for the unweighted MVC problem, the

best polynomial-time approximation algorithms have an approximation ratio of $2 - o(1)$ [41]. The MVC problem is known to be APX-hard [17, 36], and if the unique games conjecture holds, the current $(2 - o(1))$-approximation algorithms are essentially best possible [43].

In the distributed context, most prominently, the problems have been studied in the standard message passing models in graphs, in the LOCAL model and the CONGEST model [53]. In both models, the graph $G = (V, E)$ on which we want to solve some graph problem also represents the network and it is assumed that the nodes V of G can communicate with each other in synchronous rounds by exchanging messages over the edges E of G. In the LOCAL model, the size of those messages is not restricted, whereas in the CONGEST model, it is assumed that each message has to consist of at most $O(\log n)$ bits, where $n = |V|$ is the number of nodes of the network graph G. In the following discussion of existing work on distributed matching and vertex cover algorithms, we concentrate on polylogarithmic-time distributed algorithms that also work for the weighted variants of the problems.

Distributed Complexity of Weighted Matchings. While the unweighted versions of both problems can be approximated within a factor of 2 by computing a maximal matching, a little more work is needed for weighted matchings and vertex covers. The first polylogarithmic-time distributed algorithm for computing a constant approximation for the maximum weighted matching (MWM) problem was presented in [56]. This algorithm was then improved in [50] and in [49], where it is shown that a $(1/2 - \varepsilon)$-approximation for MWM can be computed in $O(\log(1/\varepsilon) \log n)$ rounds in the CONGEST model. In [7], it was further shown that can one compute a 1/2-approximation for MWM in time $O\big(\log W \cdot T_{\text{MIS}}\big)$ in the CONGEST model, where W is the ratio between the largest and smallest edge weight and where T_{MIS} is the time for computing a maximal independent set. The paper also shows that for constant ε, a $(1/2 - \varepsilon)$-approximation can be computed in only $O\big(\frac{\log \Delta}{\log \log \Delta}\big)$ rounds. Note that as shown in [47], this time complexity is best possible for any constant approximation algorithm, even in the LOCAL model. All the above algorithms are randomized. In [25], Fischer gave a deterministic CONGEST algorithm to compute a $(1/2 - \varepsilon)$-approximate weighted matching with a round complexity of $O(\log^2 \Delta \cdot \log 1/\varepsilon + \log^* n)$. This algorithm was refined in [2], where it was shown that in time $O\big(\frac{\log^2(\Delta/\varepsilon) + \log^* n}{\varepsilon} + \frac{\log(\Delta W)}{\varepsilon^2}\big)$, it is even possible to deterministically compute a $(2/3 - \varepsilon)$-approximation for the MWM problem in general graphs and a $(1 - \varepsilon)$-approximation for the MWM problem in bipartite graphs, in the CONGEST model. To the best of our knowledge, this is the only existing polylogarithmic-time CONGEST algorithm to obtain an approximation ratio that is better than 1/2. It has been observed already in [45,49,52] that in the LOCAL model, better approximations for maximum weighted matching can be computed efficiently. In particular, [49, 52] show that even in general graphs, a $(1 - \varepsilon)$-approximation can be computed in poly $\big(\frac{\log n}{\varepsilon}\big)$ rounds. It has later been shown that this can also be achieved deterministically [31]. The best known LOCAL MWM approximation algorithms are by Harris [35], who shows that a $(1 - \varepsilon)$-approximation can be computed in randomized time $\tilde{O}\big(\frac{\log \Delta}{\varepsilon^3}\big) + \text{poly}\log\big(\frac{\log \log n}{\varepsilon}\big)$ and in deterministic time $\tilde{O}\big(\frac{\log^2 \Delta}{\varepsilon^4} + \frac{\log^* n}{\varepsilon}\big)$. Those algorithms are based on computing large matchings in hypergraphs defined by paths of length $O(1/\varepsilon)$ and they unfortunately cannot directly turned into efficient CONGEST algorithms. To the best of our knowledge, even for constant $\varepsilon > 0$, the only efficient CONGEST algorithms are for the unweighted maximum matching problem. Lotker, Patt-Shamir, and Pettie [49] give an algorithm to compute a $(1 - \varepsilon)$-approximation for the *unweighted* maximum matching problem in time only $2^{O(1/\varepsilon)} \cdot \log n$ in the randomized CONGEST model. In [7] (full version), this was improved to $2^{O(1/\varepsilon)} \cdot \frac{\log \Delta}{\log \log \Delta}$. We obtain similar algorithms for the weighted matching problem. Obtaining a $(1 - \varepsilon)$-approximation

in poly $\left(\frac{\log n}{\varepsilon}\right)$ CONGEST rounds is one of the key open questions in understanding the distributed complexity of maximum matching. Fischer, Mitrović, and Uitto [26] recently settled a related problem for unweighted matchings in the streaming model and in the latest version of their paper, they even obtain a poly $\left(\frac{\log n}{\varepsilon}\right)$-round CONGEST algorithm for the unweighted matching problem.

Distributed Complexity of Weighted Vertex Covers. The first distributed constant-factor approximation algorithm for the minimum weighted vertex cover (MWVC) problem is due to Khuller, Vishkin, and Young [44]. They describe a simple deterministic algorithm to obtain a $(2+\varepsilon)$-approximation for MWVC. The algorithm can directly be implemented in $O(\log(n) \cdot \log(1/\varepsilon))$ rounds in the CONGEST model. The time for computing a $(2+\varepsilon)$-approximation has subsequently been improved to $O(\log(\Delta)/\operatorname{poly}(\varepsilon))$ in [47] and to $O(\log\Delta/\log\log\Delta)$ in [9–11] (with a very minor dependency on ε in [11]). Note that as for maximum matching, this dependency on Δ is optimal for any constant-factor approximations [46]. The algorithm of [11] can also be used to compute a 2-approximate weighted vertex cover in time $O(\log n)$. Other polylogarithmic-time algorithms to compute 2-approximations for MWVC appeared in [34, 44, 45]. In the LOCAL model, one can use generic techniques from [30, 54] (or the techniques from this paper) to deterministically compute a $(1+\varepsilon)$-approximate minimum weighted vertex cover in time poly $\left(\frac{\log n}{\varepsilon}\right)$. Further, in [32], it was shown that even on bipartite graphs with maximum degree 3, there exists a constant $\varepsilon_0 > 0$ such that computing a $(1+\varepsilon_0)$-approximate (unweighted) vertex cover requires $\Omega(\log n)$ rounds, even in the LOCAL model and even when using randomization. We generalize this result and show that for computing a $(1+\varepsilon)$-approximation, one requires $\Omega(\log(n)/\varepsilon)$ rounds. While for maximum matching, there are several CONGEST algorithms that achieve approximation ratios that are better than $1/2$, for the minimum vertex cover problem, efficiently achieving an approximation ratio significantly below 2 in general graphs might be a hard problem. In this case, computing an exact solution even has a lower bound of $\tilde{\Omega}(n^2)$ rounds in the CONGEST model and it is therefore basically as hard as any graph problem can be in this model [13]. To what extent we can achieve approximation ratios below 2 in the CONGEST model for variants of the minimum vertex cover problem is an interesting open question. There recently has been some progress. In [12], it is shown that the minimum vertex cover problem (and also the maximum matching problem) can be solved more efficiently if the optimal solution is small. In particular, if the size of an optimal vertex cover is at most k, a minimum vertex cover can be computed deterministically in time $O(k^2)$ and a $(2-\varepsilon)$-approximate solution can be computed deterministically in time $O(k+(\varepsilon k)^2)$ (and slightly more efficiently with randomization). This was the first efficient CONGEST algorithm that achieves an approximation ratio below 2 for the minimum vertex cover problem for some graphs. In [23], it was shown that in bipartite graphs, a $(1+\varepsilon)$-approximation can be computed in time poly $\left(\frac{\log n}{\varepsilon}\right)$. One of the main results of this paper is a generalization of this result to the weighted vertex cover problem. Further, it has recently been shown that on the square graph G^2, it is possible to compute a $(1+\varepsilon)$-approximate (unweighted) vertex cover in time $O(n/\varepsilon)$ in the CONGEST model (on G) [8].

1.1 Our Contributions

We next state our main contributions in detail. We prove new CONGEST upper bounds for approximating minimum weighted vertex cover and maximum weighted matching (MWM). We start by describing our results for the vertex cover problem. In [23], it was shown that in bipartite graphs, the unweighted vertex cover problem can be $(1+\varepsilon)$-approximated in poly log $\left(\frac{\log n}{\varepsilon}\right)$ time in the CONGEST model. The following theorem is a generalization of the result of [23] to weighted graphs.

▶ **Theorem 1.** *For every $\varepsilon \in (0,1]$, there is a* deterministic CONGEST *algorithm to compute a $(1+\varepsilon)$-approximation for the minimum weighted vertex cover problem in bipartite graphs in time* $\operatorname{poly}\left(\frac{\log n}{\varepsilon}\right)$.

The next theorem shows that in graph families that are closed under taking (induced) subgraphs and in which we can efficiently compute large (or heavy) independent sets, we can efficiently approximate minimum (weighted) vertex cover with an approximation ratio that is better than 2.

▶ **Theorem 2.** *Let $\mathcal{G}$ be a family of weighted graphs that is closed under taking induced subgraphs and such that for some $\lambda \in (0,1/2]$ and any n-node graph $G=(V,E,w)$ of $\mathcal{G}$, there is a $T_\lambda(n)$-round* CONGEST *algorithm to compute an independent set S of weight $w(S) \geq \lambda w(V)$. Then, there is $T_\lambda(n) + \operatorname{poly}\left(\frac{\log n}{\varepsilon}\right)$-round* CONGEST *algorithm to compute a $(2-2\lambda+\varepsilon)$-approximate weighted vertex cover for graphs of $\mathcal{G}$. If the independent set algorithm is deterministic, then also the vertex cover algorithm is deterministic.*

Note that the algorithm of Theorem 2 uses the bipartite vertex cover algorithm of Theorem 1 as a subroutine. Theorem 2 in particular implies that for graphs for which we can compute a coloring with a small number of colors, we can efficiently compute a non-trivial vertex cover approximation.

▶ **Corollary 3.** *Let $\mathcal{G}$ be a family of weighted graphs such that for some non-negative integer C, for any n-node graph $G=(V,E,w)$ of $\mathcal{G}$, there is a $T_C(n)$-round* CONGEST *algorithm to compute a vertex coloring of G with C colors. Then, there is a $T_C(n) + \operatorname{poly}\left(\frac{\log n}{\varepsilon}\right)$-round* CONGEST *algorithm to compute a $(2-2/C+\varepsilon)$-approximation of the minimum weighted vertex cover problem for graphs of $\mathcal{G}$. If the coloring algorithm is deterministic, then also the vertex cover algorithm is deterministic.*

In order to efficiently compute an independent set S of weight $w(S) \geq w(V)/C$ from a C-coloring, we need the graph to be of small diameter. However, by using standard clustering techniques (which we anyways need to apply also for our bipartite vertex cover algorithm), one can reduce the minimum (weighted) vertex cover problem on general n-node graphs to graphs of diameter $\operatorname{poly}\left(\frac{\log n}{\varepsilon}\right)$. In particular, in graphs of arboricity a, we can (deterministically) compute a $(2+\varepsilon)a$-coloring in time $O(\log^3 a \cdot \log n)$ [29]. As a consequence, we get a deterministic $\operatorname{poly}\left(\frac{\log n}{\varepsilon}\right)$-round CONGEST algorithm for computing a $(2-1/a+\varepsilon)$-approximation of minimum weighted vertex cover in graphs of arboricity a.

In addition to our CONGEST algorithms for approximating minimum weighted vertex cover, we also provide new CONGEST algorithms for approximating maximum weighted matching. The following theorem can be seen as a generalization of Theorem 3.15 in [49] and of Theorem B.12 in [7] (full version).

▶ **Theorem 4.** *For every $\varepsilon, \delta \in (0,1]$, there is a* randomized CONGEST *algorithm that with probability at least $1-\delta$ computes a $(1-\varepsilon)$-approximation to the maximum weighted matching problem in $2^{O(1/\varepsilon)} \cdot \left(\log(W\Delta) + \log^2 \Delta + \log^* n\right) \cdot \log^3(1/\delta)$ rounds. Further, there is a* deterministic CONGEST *algorithm to compute a $(1-\varepsilon)$-approximation for the minimum weighted matching problem in time $2^{O(1/\varepsilon)} \cdot \operatorname{poly}\log n$.*

Note that except for the $\log^* n$ term, for constant error probability δ, the round complexity of our randomized algorithm is independent of the number of nodes n. Moreover, for constant ε and δ, in bounded-degree graphs with bounded weights, the round complexity of the randomized algorithm is only $O(\log^* n)$. For unweighted matchings, a round complexity that is completely independent of n was obtained by [7]. Interestingly, Göös and Suomela in [32]

showed that such a result is not possible for the minimum vertex cover problem, even in the LOCAL model. They show that even for bipartite graphs of maximum degree 3, there exists a contant $\varepsilon_0 > 0$ such that any randomized distributed $(1+\varepsilon_0)$-approximation algorithm for the (unweighted) minimum vertex cover problem requires $\Omega(\log n)$ rounds. As our last contribution, we generalize the result of [32] to computing $(1+\varepsilon)$-approximate solutions for any sufficiently small $\varepsilon > 0$.

▶ **Theorem 5.** *There exists a constant $\varepsilon_0 > 0$ such that for every $\varepsilon \in (0, \varepsilon_0]$, any randomized* LOCAL *model algorithm to compute a $(1+\varepsilon)$-approximation for the (unweighted) minimum vertex cover problem in bipartite graphs of maximum degree 3 requires $\Omega\big(\frac{\log n}{\varepsilon}\big)$ rounds.*

Theorem 5 is obtained by a relatively simple reduction to the $(1+\varepsilon_0)$-approximation lower bound proven in [32] for bipartite graphs of maximum degree 3. We note that if we only require the approximation factor to hold in expectation, as discussed at the end of Section 3, in the LOCAL model the theorem is tight even for general graphs and even for the weighted vertex cover problem.

Organization of the paper. The remainder of the paper is organized as follows. In Section 2, we define the communication model and we introduce all the necessary mathematical notations and definitions. In Section 4, we give an overview over our minimum weighted vertex cover algorithms and in Section 5, we discuss the most important ideas to derive our maximum weighted matching result. Many of the technical details regarding our algorithms have to be omitted in this extended abstract and they only appear in the full version [22] of this paper. In Section 6, we prove the lower bound on approximating vertex cover in bipartite graphs in the LOCAL model. Finally, in Appendix A, we give some basic algorithmic tools that we need for our algorithms.

2 Model and Preliminaries

2.1 Mathematical Notation

Let $G = (V, E, w)$ be an undirected weighted graph, where w is a non-negative weight function. We will use node and edge weights in the paper and depending on the context, we will use w to assigns weights to nodes and/or edges. Generally for a set X of nodes and/or edges, we use $w(X)$ to denote the sum of the weights of all nodes/edges in X. For example, if we have node weights, $w(V)$ denotes the sum of the weights of all the nodes. Throughout the paper, we assume that all weights are integers that are polynomially bounded in the number of node of the graph. However, as long as we can communicate a single weight in a single message, all our algorithms can be adapted to also work at no significant additional asymptotic cost for more general weight assignments. We further use the following notation for graphs. For a node $v \in V$, we use $N(v) \subseteq V$ to denote the set of neighbors of v and we use $E(v) \subseteq E$ to denote the set of edges that are incident to v.

For a graph $G = (V, E)$, the *bipartite double cover* is defined as the graph $G_2 := G \times K_2 = (V \times \{0,1\}, E_2)$, where there is an edge between two nodes (u, i) and (v, j) in E_2 if and only if $\{u, v\} \in E$ and $i \neq j$. Hence, in G_2, every node u of G is replaced by two nodes $(u, 0)$ and $(u, 1)$ and every edge $\{u, v\}$ of G is replaced by the two edges $\{(u, 0), (v, 1)\}$ and $\{(u, 1), (v, 0)\}$. Moreover, if G is a weighted graph with weight function w, we assume that the bipartite double cover G_2 is also weighted and that the corresponding nodes and/or edges have the same weight as in G. That is, in case of node weights, for every $u \in V$, we define $w((u, 0)) = w((u, 1)) = w(u)$ and in case of edge weights, for every $\{u, v\} \in E$, we define $w(\{u, i\}, \{v, 1-i\}) = w(\{u, v\})$ for $i \in \{0, 1\}$.

2.2 Problem Definitions

In this paper, we consider the *minimum weighted vertex cover (MWVC)* and the *maximum weighted matching (MWM)* problems. Formally, in the MWVC problem, we are given a weighted graph $G = (V, E, w)$ with positive node weights. A vertex cover of G is a set $S \subseteq V$ of nodes such that for every edge $\{u, v\} \in E$, $S \cap \{u, v\} \neq \emptyset$. The goal of the MWVC problem is to find a vertex cover S of minimum total weight $w(S)$. In the MWM problem, we are given a weighted graph $G = (V, E, w)$ with positive edge weights. A matching of G is a set $M \subseteq E$ of edges such that no two edges in M are adjacent. The goal of the MWM problem is to find a matching M of maximum total weight $w(M)$. The unweighted versions of the two problems are closely related to each other as their natural fractional linear programming (LP) relaxations are duals of each other. In the paper, we will also use the fractional relaxation of the MWVC problem and its dual problem. In the fractional MWVC problem on G, every node $u \in V$ is assigned a value $x_u \in [0, 1]$ such that for every edge $\{u, v\} \in E$, $x_u + x_v \geq 1$ and such that the sum $\sum_{u \in V} w(u) \cdot x_u$ is minimized. The dual LP of this problem, which for a given weight function w, we in the following call the *fractional w-matching* problem, is defined as follows. Every edge $e \in E$ is assigned a (fractional) value $y_e \geq 0$ such that for every node $u \in V$, we have $\sum_{e:u \cap e \neq \emptyset} y_e \leq w(u)$ and such that the sum $\sum_{e \in E} y_e$ is maximized. We use the vector $\boldsymbol{y}$ to refer to a fractional solution that assigns a fractional value y_e to every edge. Further for convenience, for a set of edges F, we also use the short notation $y(F) := \sum_{e \in F} y_e$. LP duality directly implies that the value of any fractional w-matching cannot be larger than the weight of any vertex cover:

▶ **Lemma 6.** *Let $G = (V, E, w)$ be a node-weighted graph and let $\boldsymbol{y}$ be a fractional w-matching of G. It then holds that $y(E) \leq w(S)$ for every vertex cover S of G.*

Proof. We have

$$w(S) = \sum_{v \in S} w(v) \geq \sum_{v \in S} \sum_{e: v \in e} y_e \geq \sum_{e \in E} y_e = y(E).$$

The first inequality holds because the values y_e form a valid fractional w-matching and the second inequality holds because S is a vertex cover. ◀

The *approximation ratio* of an approximation algorithm for the MWVC or MWM problem is defined as the worst-case ratio between the total weight of a vertex cover or matching computed by the algorithm over the total weight of an optimal vertex cover or matching. That is, we define the approximation ratio such that it is ≥ 1 for minimization and ≤ 1 for maximization problems.

2.3 Low-Diameter Clustering

Many of our algorithms have some components that require global communication in the network. In order the achieve a polylogarithmic round complexity, we therefore need a graph with polylogarithmic diameter. We achieve this by applying standard clustering techniques. Formally, we use the clusterings as in [23] described in the following. Let $G = (V, E, w)$ be a weighted graph with non-negative node and edge weights. A *clustering* of G is a collection $\{S_1, \ldots, S_k\}$ of disjoint node sets $S_i \subseteq V$. For $\lambda \in [0, 1]$, a clustering $\{S_1, \ldots, S_k\}$ is called *λ-dense* if the total weight of all nodes and edges in the induced subgraphs $G[S_i]$ for $i \in \{1, \ldots, k\}$ is at least $\lambda(w(V) + w(E))$. Further, for an integer $h \geq 1$, a clustering $\{S_1, \ldots, S_k\}$ is called *h-hop separated* if for any two clusters S_i and S_j $(i \neq j)$ and any pair of nodes $(u, v) \in S_i \times S_j$, we have $d_G(u, v) \geq h$, where $d_G(u, v)$ denotes the hop-distance

between u and v. Further for two integers $c, d \geq 1$, a clustering $\{S_1, \ldots, S_k\}$ is defined to be *(c,d)-routable* if we are given a collection of $T_1, \ldots, T_k$ trees in G such that for each $i \in \{1, \ldots, k\}$, the nodes S_i are contained in T_i, every tree T_i has diameter at most d, and every edge $e \in E$ of G is contained in at most c of the trees $T_1, \ldots, T_k$. Note that this implies that each cluster of a (c, d)-routable clustering has weak diameter at most d and if the nodes of T_i are all contained in S_i, it implies that the strong diameter of cluster S_i is at most d.

2.4 Communication Model

Throughout the paper, we assume a standard synchronous message passing model on graphs. That is, the network is modeled as an undirected n-node graph $G = (V, E)$. Each node is equipped with a unique $O(\log n)$-bit identifier. The nodes V communicate in synchronous rounds over the edges E such that in each round, every node can send an arbitrary mesage to each of its neighbors. Internal computations at the nodes are free. Initially, the nodes do not know anything about the topology of the network. When computing a vertex cover or a matching, at the end of the algorithm, every node must output if it is in the vertex cover or which of its edges belong to the matching. The time or round complexity of an algorithm is defined as the number of rounds that are needed until all nodes terminate. If the size of the messages is not restricted, this model is known as the LOCAL model [53]. In the more restrictive CONGEST model, all messages must consist of at most $O(\log n)$ bits [53]. In several of our algorithms, we will first compute a clustering as defined above in Section 2.3 and we afterwards run CONGEST algorithms on the clusters. If we are given a (c, d)-routable clustering, we are only guaranteed that the diameter of each cluster S_i is small if we add the nodes and edges of the tree T_i to the cluster. For running our algorithms on individual clusters, we therefore need an extension of the classic CONGEST model, which has been introduced as the SUPPORTED CONGEST model in [27,55]. In the SUPPORTED CONGEST model, we are given two graphs, a communication graph $H = (V_H, E_H)$ and a logical graph $G = (V, E)$, which is a subgraph of H. When solving a graph problem such as MWVC or MWM in the SUPPORTED CONGEST model, we need to solve the graph problem on the logical graph G, we can however use CONGEST algorithms on the underlying communication graph H to do so. Note that if we are given a (c, d)-routable clustering, we can define $G_i := G[S_i]$ and H_i as the union of the graph G_i and the tree T_i for each cluster and we can then in parallel run 1 round of a SUPPORTED CONGEST algorithm on each cluster in c CONGEST rounds on G.

3 Reducing to Small Diameter

We start the presentation of our results by describing how we can use clusterings to reduce the problem of approximating MWVC or MWM on general graphs to the case of approximating the same problems on graphs of small diameter. The high-level idea that we use to reduce the diameter is a classic one. We find a disjoint and sufficiently separated collection of low-diameter clusters such that only a small fraction of the graph is outside of the clusters (see, e.g., [6, 48, 51, 53, 54] for constructions of such clusterings). We then compute a good approximation for a given problem inside each cluster and we use a coarse approximation to extend the solution to the parts of the graph outside of the clusters. If we want this to work in general graphs, we have to adapt the standard clustering constructions such that the part of the graph that is outside of the clusters contains only a *small fraction of a solution to the actual problem* that we want to approximate, rather than simply a small fraction of the number of nodes and/or edges of the graph. A generic way to achieve this in the LOCAL

model has been described in [30] and a method that can also be used in the CONGEST model has recently been described in [23] for the unweighted minimum vertex cover problem. The following theorem shows how to extend the approach of [23] to work also for weighted vertex cover and matching. The theorem shows that at the cost of a $(1+\varepsilon)$-factor, the problems of approximating MWVC and MWM can efficiently be reduced to approximating the problems in the SUPPORTED CONGEST model with a small-diameter communication graph. We here use the SUPPORTED CONGEST model because we compute clusters by adapt a network decomposition algorithm of [54], which only creates clusters of weak diameter. Due to lack of space, the proof of the following theorem is omitted and deferred to the full version of this paper [22].

▶ **Theorem 7** (Diameter Reduction). *Let $T^{\alpha}_{\mathsf{SC}}(n,D)$ be the time required for computing an α-approximation for the MWVC or the MWM problem in the* SUPPORTED CONGEST *model with a communication graph of diameter D. Then, for every $\varepsilon \in (0,1]$, there is a* poly $\big(\frac{\log n}{\varepsilon}\big) + O\big(\log n \cdot T^{\alpha}_{\mathsf{SC}}\big(n, O\big(\frac{\log^3 n}{\varepsilon}\big)\big)\big)$*-round* CONGEST *algorithm to compute a $(1-\varepsilon)\alpha$-approximation of MWM or an $(1+\varepsilon)\alpha$-approximation of MWVC in the* CONGEST *model. If the given* SUPPORTED CONGEST *model algorithm is deterministic, then the resulting* CONGEST *model algorithm is also deterministic. Also, if we want to solve MWVC or MWM in the* CONGEST *model on a bipartite graph, then it is sufficient to have a* SUPPORTED CONGEST *model algorithm that works for a bipartite communication (and thus also logical) graph.*

▶ Remark. We note that by using a variant of the randomized clustering algorithm of Miller, Peng, and Xu [51], one can compute a $\big(1, O\big(\frac{\log n}{\varepsilon}\big)\big)$-routable, 3-hop separated clustering with expected density $1-\varepsilon$ in time $O\big(\frac{\log n}{\varepsilon}\big)$ (also cf. [23]). In combination with the argument in the above proof, this in particular implies that it in $O\big(\frac{\log n}{\varepsilon}\big)$ rounds in the LOCAL model, it is possible to compute weighted matchings and weighted vertex covers with expected approximation ratios $1-\varepsilon$ and $1+\varepsilon$, respectively.

4 Weighted Vertex Cover Algorithms

We next describe our results on approximating the minimum weighted vertex cover. We start by describing a distributed approximation scheme for bipartite graphs.

4.1 Basic Weighted Vertex Cover Algorithm

It is well-known that in bipartite graphs, the size of a maximum matching is equal to the size of a minimum vertex cover (this is known as Kőnig's theorem [16, 42]). The theorem was also independently discovered in [20] by Egerváry, who also more generally proved that on node-weighted bipartite graphs, the total value of an optimal (fractional) w-matching is equal to the weight of a minimum weighted vertex cover, where a fractional w-matching of a node-weighted graph $G=(V,E,w)$ is an assignment of fractional values $y_e \geq 0$ to all edges such that the edges of each node v sum up to at most $w(v)$. In both cases, the theorem can be proven in a constructive way. Given a maximum matching or more generally a maximum fractional w-matching, there is a simple (and efficient) algorithm to compute a vertex cover of the same size or weight.

Moreover as shown in [24], if we are given a good approximate matching or w-matching with some additional properties, the constructive proof of [20, 42] can be adapted to obtain a good approximate (weighted) vertex cover. For the unweighted case, this method is at the core of the CONGEST model bipartite vertex cover algorithms of [23]. We next describe how to use this technique to approximate MWVC in the CONGEST model.

Let $G = (V, E, w)$ be a node-weighted graph, where w is a non-negative node weight function. For a node $v \in V$ and a given fractional w-matching y_e for $e \in E$, we define the *slack* of v as $s(v) := w(v) - \sum_{e:v\in e} y_e$. Given a fractional w-matching y_e for $e \in E$, an *augmenting path* is an odd length path $P = (v_0, \dots, v_{2k+1})$ where the end nodes v_0 and v_{2k+1} have positive slack $s(v_0), s(v_{2k+1}) > 0$, and for $i \in \{1, 2, ..., k\}$, each even edge $e = (v_{2i-1}, v_{2i})$ has a positive fractional value $y_e > 0$. Assume that we are given a bipartite graph $G = (A \cup B, E, w)$ and that we are given a fractional w-matching $\boldsymbol{y}$ of G such that there are no augmenting paths of length at most $2k - 1$ for some integer $k \geq 1$. We can then apply the following algorithm to compute a vertex cover S of G. The algorithm computes disjoint sets $A_0, A_1, \dots, A_k \subseteq A$ and disjoint sets $B_1, \dots, B_k \subseteq B$. In the following for a set of nodes X, we use $N_{y>0}(X)$ to denote the set of nodes that are connected to a node in X through an edge e with $y_e > 0$.

Algorithm 1 Basic Approximate Weighted Vertex Cover Algorithm.

1. Define $A_0 := \{v \in A : s(v) > 0\}$ as the nodes in A with positive slack and $B_0 := \emptyset$.
2. For every $i \in \{1, \dots, k\}$, define $B_i := \left\{v \in B \setminus \bigcup_{j=0}^{i-1} B_j : v \in N(A_{i-1})\right\}$.
3. For every $i \in \{1, \dots, k\}$, define $A_i := \left\{v \in A \setminus \bigcup_{j=0}^{i-1} A_j : v \in N_{y>0}(B_i)\right\}$.
4. Define $i^* := \arg\min_{i\in\{1,\dots,k\}} w(B_i)$.
5. Output $S := \bigcup_{i=1}^{i^*} B_i \cup \left(A \setminus \bigcup_{i=0}^{i^*-1} A_i\right)$.

That is, the sets $A_0, B_1, A_1, B_2, A_2 \dots$ are the levels of a BFS traversal of the graph starting at the nodes in A_0 and where steps from B_i to A_i have to be over an edge e with positive fractional value $y_e > 0$.

▶ **Lemma 8.** *Given a weighted bipartite graph $G = (A \cup B, E, w)$, an integer $k \geq 1$, and a fractional w-matching of G with no augmenting paths of length at most $2k-1$, the above algorithm computes a $(1 + 1/k)$-approximate weighted vertex cover of G. Further, the above algorithm can be deterministically implemented in $O(D + k)$ rounds in the* SUPPORTED CONGEST *model if the communication graph is also bipartite and has diameter at most D.*

Proof. We first prove that S is a valid vertex cover. For this, we need to show that there is no edge between $A \setminus S$ and $B \setminus S$. We have $A \setminus S = \bigcup_{j=0}^{i^*-1} A_j$ and $B \setminus S = \bigcup_{j=i^*+1}^{k} B_j \cup \overline{B}$, where $\overline{B} = B \setminus \bigcup_{i=1}^{k} B_i$. We therefore need to show that there cannot be an edge between a set A_j for $j < i^*$ and $\sum_{j=i^*+1}^{k} B_j \cup \overline{B}$. However, by construction, all neighbors of nodes in A_j are in $B_1, \dots, B_{i^*}$ and we can thus conclude that S is a vertex cover.

We next show that S is a $(1 + 1/k)$-approximate weighted vertex cover of G. First observe that for all $i \in \{1, 2.., k\}$, all nodes v in B_i are saturated nodes with zero slack, i.e., $w(v) = \sum_{e\in E(v)} y_e$. From the construction of the sets $A_0, B_1, A_1, \dots$, we otherwise get an augmenting path of length at most $2k - 1$. Moreover since the sets of edges incident to the sets $B_1, \dots, B_k$ are disjoint, the sum of the fractional values of all those edges is at most $y(E) = \sum_{e\in E} y_e$. By the choice of i^*, we therefore know $w(B_{i^*}) = \sum_{v\in B_{i^*}} \sum_{e\in E(v)} y_e \leq y(E)/k$. Moreover, from the BFS construction of the sets A_i and B_i it follows that the only edges e with $y_e > 0$ for which both nodes are in S are edges that are incident to nodes in B_i^*. We can therefore conclude that $w(S) \leq y(E) + w(B_{i^*}) \leq (1 + 1/k) \cdot y(E)$. The approximation ratio now follows from LP duality (i.e., from Lemma 6).

Finally, we discuss how the above algorithm can be efficiently implemented in time $O(D + k)$ rounds in the SUPPORTED CONGEST model, where D is the diameter of the communication graph H. In $O(D)$ rounds, one can compute a BFS spanning tree of H and use it to compute the bipartition of the nodes of H and thus of G into sets A and B. Then in $O(k)$ rounds, the algorithm constructs the sets A_i and B_i for $i \in \{1, 2.., k\}$ by running the first $2k$ iterations of parallel BFS on G starting from set A_0 (where edges from B_i to A_i need to have positive fractional values). Finally in another $O(D + k)$ rounds, we use the precomputed BFS spanning tree on H and a standard pipelining scheme for the root to compute the weights of all the sets B_i, determine the index i^* of the smallest weight amongst them, and broadcast it to all nodes in G. ◀

We remark that although the above algorithm only requires a BFS traversal from all nodes in A_0 for k levels, the algorithm still requires time D for two reasons. First, the algorithm needs to know the bipartition $A \cup B$ of G and computing the bipartition requires $\Omega(D)$ time. Second, even if the bipartition is given initially, the algorithm still needs $\Omega(D)$ time to determine the optimal level i^*.

4.2 Getting Rid of Short Augmenting Paths

The basic MWVC algorithm described in Section 4.1 basically converts a good approximation of fractional w-matching into a good MWVC approximation. However the algorithm needs a fractional w-matching with the additional property that there are no short augmenting paths. For the unweighted setting, there exists a randomized CONGEST algorithm to compute an integral matching with no short augmenting paths [49]. It is however not clear if the algorithm of [49] can be generalized to the fractional w-matching problem. Further, even in the unweighted case, we do not have a deterministic CONGEST algorithm to compute such a matching. As in the deterministic, unweighted MVC algorithm of [23], we therefore use a different approach. In the unweighted setting, we first compute a $(1 - \delta)$-approximate matching that can potentially have short augmenting paths. We then get rid of those short augmenting paths by removing at least one unmatched node or both nodes of a matching edge from the graph. The removed nodes are at the end added to the vertex cover to make sure that all edges are covered. The selection of a smallest possible number of unmatched nodes and matching edges that hit all short augmenting paths can be phrased as a minimum set cover problem, which we can approximate efficiently in the CONGEST model. In the weighted case, we use a generalization of this approach. Because of the weights, the process and its analysis however becomes more subtle and we have to be more careful.

Assume that we want to compute a $(1 + O(\varepsilon))$-approximate weighted vertex cover for a node-weighted bipartite graph $G = (A \cup B, E, w)$. In a first step, we compute a $(1 - \delta)$-approximate fractional w-matching $\boldsymbol{y} := \{y_e : e \in E\}$ of G for some parameter $\delta \ll \varepsilon$. We can do this efficiently by using Theorem 12. The fractional w-matching $\boldsymbol{y}$ however might have short augmenting paths. In a second step, we then convert our graph G and the fractional w-matching $\boldsymbol{y}$ such that we obtain an instance with no short augmenting paths and that we can thus apply Lemma 8. More concretely, we decrease some of the weights $w(v)$ and some of the fractional values y_e such that for the resulting weights $w'(v)$ and the resulting w'-matching $\boldsymbol{y}'$, the graph G has no short augmenting paths and such that a $(1+\varepsilon)$-approximate weighted vertex cover of G with the weights $w'(v)$ is a $(1 + O(\varepsilon))$-approximate weighted vertex cover of G for the original weights. We next describe the main ideas of this transformation.

Formally, the conversion can be defined by a set $X \subseteq \{v \in A \cup B : s(v) > 0\}$ of nodes with positive slack and a set $F \subseteq \{e \in E : y_e > 0\}$ of edges with positive fractional value. The new fractional values y'_e and the new weights $w'(v)$ are defined as follows:

$$y'_e := \begin{cases} 0 & \text{if } e \in F \\ y_e & \text{if } e \notin F \end{cases}, \qquad w'(v) := \begin{cases} w(v) - s(v) - y(E(v) \cap F) & \text{if } v \in X \\ w(v) - y(E(v) \cap F) & \text{if } v \notin X \end{cases} \tag{1}$$

▶ **Lemma 9.** *Any augmenting path of G w.r.t. the weight function w' and the fractional w'-matching $\boldsymbol{y}'$ is also an augmenting path w.r.t. the original weight function w and the original fractional w-matching $\boldsymbol{y}$.*

Proof. First note that whenever we decrease a value y_e to $y'_e < y_e$ for some edge $e = \{u, v\}$, we also decrease the weights of u and v by the same amount. Therefore, the slack of a node v w.r.t. w' and $\boldsymbol{y}'$ cannot be larger than the slack of v w.r.t. w and $\boldsymbol{y}$. Therefore any odd-length path P in G that starts and ends at a node with positive slack w.r.t. w' and $\boldsymbol{y}'$ also starts and ends at a node with positive slack w.r.t. w and $\boldsymbol{y}$. Further, if every even edge e of such a path P has a positive fractional value $y'_e > 0$, then it also holds that $y_e > 0$. ◀

Note that the definition of w' and $\boldsymbol{y}'$ in (1) guarantees that all nodes $v \in X$ have slack 0 w.r.t. the new weights w' and the new fractional values $\boldsymbol{y}'$. Consider some augmenting path $P = (v_0, \dots, v_{2\ell+1})$ of G w.r.t. w and $\boldsymbol{y}$. If we have $v_0 \in X$ or $v_{2\ell+1} \in X$ or if we have $e \in F$ for one of the even edges $\{v_{2i-1}, v_{2i}\}$ of P, then P is not an augmenting path of G w.r.t. w' and $\boldsymbol{y}'$. In order to get rid of all short augmenting paths, we therefore need to choose one of the end nodes or one of the even edges of each such path and add them to X or F. We can then use Lemma 8 to efficiently compute a good vertex cover approximation for G w.r.t. the new weights w'. The quality of such a vertex cover w.r.t. the original weights w can be bounded as follows.

▶ **Lemma 10.** *Let S^* be an optimal weighted vertex cover of $G = (V, E)$ w.r.t. the weights w and assume that for some $\alpha \geq 1$, S is an α-approximate weighted vertex cover of G w.r.t. the weights w'. It then holds that*

$$w(S) \leq \alpha \cdot w(S^*) + s(X) + y(F), \qquad \text{where } s(X) := \sum_{v \in X} s(v).$$

Proof. Let S' be an optimal weighted vertex cover of G w.r.t. the weights w'. Because any vertex cover must contain at least one node of every edge in F, we have $w'(S') \leq w(S^*) - y(F)$. We therefore have

$$w(S) \leq w'(S) + s(X) + 2y(F) \leq \alpha w'(S') + s(X) + 2y(F) \leq \alpha w(S^*) + s(X) + y(F). \quad ◀$$

In order to optimize the approximation, we thus need to determine the sets X and F such that $s(X) + y(F)$ is as small as possible and such that we "cover" all short augmenting paths. The problem of finding the best possible sets X and F can naturally be phrased as a weighted set cover problem. One can further show that if the parameter δ that determines the quality of the fractional w-matching $\boldsymbol{y}$ is chosen sufficiently small (but still as $\delta = \text{poly}(\varepsilon / \log n)$), even a logarithmic approximation to this weighted set cover instance guarantees that $s(X) + y(F) = O(\varepsilon \cdot w(S^*))$. Further, by sequentially going over the possible short augmenting path lengths and adapting existing algorithms of [49] and [23], a variant of the greedy algorithm for this weighted set cover instance can be implemented efficiently in the CONGEST model on G. Given an efficient algorithm to find appropriate sets X and F, the claim of Theorem 1 then follows almost immediately by combining with Theorem 7 and Lemma 8. The technical details appear in the full version [22].

4.3 Generalization to Non-Bipartite Graphs

For approximating the MWVC problem in general graphs, we employ a standard approach that is for example described in [37]. We describe the details for completeness. The minimum fractional (weighted) vertex cover problem is the natural LP relaxation of the minimum (weighted) vertex cover problem. That is, a fractional vertex cover of a graph $G = (V, E)$ is an assignment of values $x_v \in [0, 1]$ to all nodes such that for every edge $\{u, v\}$, $x_u + x_v \geq 1$. While the integrality gap of the (weighted and unweighted) vertex cover can be arbitrarily close to 2, it is well-known that there are optimal fractional solutions that are *half-integral* That is, there are optimal fractional solutions such that for all nodes v, $x_v \in \{0, 1/2, 1\}$. Given such a fractional solution, let S_x be the set of nodes v with $x_v = x$ for $x \in \{0, 1/2, 1\}$. Let $I_{1/2}$ be an independent set of the induced subgraph $G[S_{1/2}]$ of the half-integral nodes. It is not hard to see that $S := S_1 \cup S_{1/2} \setminus I_{1/2}$ is a vertex cover of G and if $w(I_{1/2}) \geq \lambda \cdot w(S_{1/2})$, then the set S is a $(2 - 2\lambda)$-approximate solution for the MWVC problem on G with weights w. If the half-integral fractional solution is only an α-approximate fractional weighted vertex cover, the resulting approximation is $(2 - 2\lambda) \cdot \alpha$.

The core to proving Theorem 2 is therefore to first compute a $(1 + \varepsilon)$-approximate half-integral fractional solution for a given weighted graph $G = (V, E, w)$. The following lemma uses a standard approach to achieve this by first computing an approximate solution to the (integral) MWVC problem for the bipartite double cover G_2 of G (see definition in Section 2).

▶ **Lemma 11.** *Let $G = (V, E, w)$ be a weighted graph and let $G_2 = (V_2, E_2)$ be the bipartite double cover of G, where each node $(v, i) \in V_2$ for $v \in V$ gets assigned weight $w(v)$. Let S be a vertex cover of G_2 and define $x_v := |\{(v, 0), (v, 1)\} \cap S|/2$ for every $v \in V$. If S is an α-approximate weighted vertex cover of G_2 for some $\alpha \geq 1$, then $\boldsymbol{x} = \{x_v : v \in V\}$ is a half-integral α-approximate fractional weighted vertex cover of G.*

Proof. We first show that the weight $\sum_{v \in V} w(v) \cdot z_v$ of an optimal fractional weighted vertex cover $\boldsymbol{z}$ of $G = (V, E, w)$ is exactly half the weight of an optimal fractional weighted vertex cover of G_2 with weights assigned as defined by the claim of the lemma. To see this, let $\boldsymbol{z}$ be a fractional vertex cover of G. We can then get a valid fractional vertex cover of G_2 by setting $z_{(v,0)} = z_{(v,1)} = z_v$ for every node $v \in V$ of G and the corresponding nodes $(v, 0)$ and $(v, 1)$ in G_2. In the other direction, for a fractional vertex cover $\boldsymbol{z}$ in G_2, we obtain a fractional vertex cover of G by setting $z_v := (z_{(u,0)} + z_{(u,1)})/2$. Note that because G_2 is a bipartite graph, an optimal (integral) weighted vertex cover of G_2 is also an optimal fractional weighted vertex cover of G_2 and therefore the vertex cover S is an α-approximate fractional weighted vertex cover of G_2. The fractional vertex cover $\boldsymbol{x}$ of G as given by the lemma statement is of half the weight of S in G_2 and it therefore is an α-approximate fractional weighted vertex cover of G. Clearly, $\boldsymbol{x}$ is half-integral. ◀

We can now prove Theorem 2 and Corollary 3.

Proof of Theorem 2. As discussed, given a weighted graph $G = (V, E, w)$, we first construct the bipartite double cover $G_2 = (V_2, E_2, w)$, where the weight function w is extended to G_2 in the obvious way (and as described in Section 4.3). Note that since every node of G only needs to simulate 2 nodes in G_2 and every edge of G is replaced by 2 edges in G_2, CONGEST algorithms on G_2 can be simulated on G with only constant overhead. By using Theorem 1, we can therefore compute a $(1 + \varepsilon)$-approximation of MWVC on G_2 in time poly $\left(\frac{\log n}{\varepsilon}\right)$. By Lemma 11, this can directly be turned into a half-integral $(1 + \varepsilon)$-approximate fractional weighted vertex cover of G. Let S_1 be the nodes of G that have a fractional vertex cover

value of 1 and let $S_{1/2}$ be the nodes of G that have a fractional vertex cover value of $1/2$. Assume further that $I_{1/2}$ is an independent set of the induced subgraph $G[S_{1/2}]$. Clearly, $S := S_1 \cup S_{1/2} \setminus I_{1/2}$ is a vertex cover of G. If the weight $w(I_{1/2})$ is $w(I_{1/2}) \geq \lambda \cdot w(S_{1/2})$, then the weight of the vertex cover S can be bounded as

$$\begin{aligned} w(S) &= w(S_1) + w(S_{1/2}) - w(I_{1/2}) \\ &\leq w(S_1) + (1-\lambda) \cdot w(S_{1/2}) \\ &\leq (2-2\lambda) \cdot \left(w(S_1) + \frac{1}{2} \cdot w(S_{1/2}) \right) \\ &\leq (2-2\lambda) \cdot (1+\varepsilon) \cdot w(S^*), \end{aligned}$$

where S^* is an optimal weighted vertex cover of G. The claim of the theorem now follows. ◀

Proof of Corollary 3. By Theorem 7, we can first reduce the diameter of the communication graph in time poly $\left(\frac{\log n}{\varepsilon}\right)$ while only losing a $(1+\varepsilon/2)$-factor in the approximation. Now assume that we are given a weighted graph G with a C-coloring, that we have a communication graph H of diameter D, and that we can use the SUPPORTED CONGEST model. In time $O(D)$ in the SUPPORTED CONGEST model, we can then use the C-coloring to compute an independent set of weight at least $w(V(G))/C$ by just keeping the heaviest color class. The corollary therefore follows directly by combining Theorem 7 and Theorem 2. ◀

5 Weighted Matching Approximation

We provide a randomized and a deterministic CONGEST algorithm to approximate the MWM problem. Both algorithms are based on the following key idea that was developed for the unweighted maximum matching problem by Lotker, Patt-Shamir, and Pettie [49] and that was extended by Bar-Yehuda et al. [7]. We iteratively adapt an initial matching M_0 of a given weighted graph $G = (V, E, w)$ as follows. We repeatedly sample bipartite subgraphs of G and we then find a good matching on this bipartition to improve the matching of G. In [7,49], the matching is improved by finding short augmenting paths in the sampled bipartite subgraph and augmenting the existing matching along those paths. While, as discussed below, also for maximum weighted matching, it is in principle possible to find augmenting paths and cycles, in the weighted case, we do not know how to do this efficiently in the CONGEST model. Instead, we use the bipartite MWM approximation algorithm of [2] to find a good matching in each sampled bipartite graph. Unlike when using augmenting paths, this approach can potentially also lead to a worse matching (if the existing matching is already a very good matching of the sampled bipartite graph). We will however see that when computing a sufficiently good approximation in each sampled bipartition, we can use the approach to improve the matching of G sufficiently often.

Before explaining our algorithms in more detail, we need to introduce the notion of augmenting paths and cycles for weighted matchings. Given a matching M, a path or cycle in which the edges alternate between edges $\in M$ and edges $\notin M$ is called an *alternating path or cycle* w.r.t. matching M. An alternating path or cycle is called an *augmenting path or cycle* w.r.t. M if swapping the matching edges with the non-matching edges increases the weight of the matching. This increase is termed the *gain* of an augmenting path/cycle w.r.t. M (we will omit the qualification 'w.r.t.' if it is clear from the context). By definition, the gain of an augmenting path is positive. Note that alternating cycles (and therefore also augmenting cycles) are always of even length.

Let M^* be a maximum weighted matching in G. Consider the symmetric difference $F = M^* \triangle M$ of M^* and some arbitrary matching M. The set F consists of (vertex-disjoint) alternating paths and cycles where the edges alternate between M and M^*. All of those alternating paths and cycles are either augmenting paths/cycles, or their M-edges have exactly the same weight as their M^*-edges. The total gain of all the (vertex-disjoint) augmenting paths and cycles induced by F is therefore exactly $w(M^*) - w(M)$.

Dealing with augmenting paths and cycles in the context of edge-weighted graphs is much more challenging than in the unweighted case. Every augmenting path in unweighted graphs improves the matching by exactly one and augmenting cycles do not exist. Further, the classic results of Hopcraft and Karp [39] imply that if the shortest augmenting path is of length ℓ, augmenting over an augmenting path of length ℓ cannot create new augmenting paths of length $\leq \ell$ and after augmenting over a maximal set of vertex-disjoint augmenting paths of length ℓ, one gets a matching with a shortest augmenting path length $\geq \ell + 2$. If in some matching M, the shortest augmenting path has length $\ell = 2k - 1 \geq 1$, we further know that M already is a $(1 - \frac{1}{k})$-approximation of the optimal matching. The $(1 - \varepsilon)$-approximation algorithm for unweighted matching in [49] heavily relies on all those properties.

Some of the properties of augmenting paths for unweighted matchings also carry over to the weighted case. In the full version [22] we show that there exists a set of vertex-disjoint augmenting paths and cycles of length at most ℓ for some $\ell = O(1/\varepsilon)$ such that augmenting over all those paths/cycles improves the current matching by at least $\frac{\varepsilon}{4} \cdot w(M^*)$. Basically, the existence of this collection of short augmenting paths can be proven by breaking long augmenting paths and cycles in the symmetric difference $F = M \triangle M^*$ into short augmenting paths. However, while in the unweighted case, a large set of vertex-disjoint short augmenting paths can be computed efficiently in the CONGEST model (see [7, 49]), it is not clear how to efficiently compute such a set of augmenting paths/cycles for weighted matchings in the CONGEST model, even in bipartite graphs (there are efficient algorithms in the LOCAL model and this has also been exploited in the literature, e.g., in [31, 49, 52]). Fortunately, there still is an efficient CONGEST algorithm for computing a $(1 - \varepsilon)$-approximate weighted matching in bipartite graphs [2]. Unlike the existing CONGEST algorithms that are based on the Hopkroft/Karp framework, the algorithm of [2] is even deterministic. It is however not based on augmenting along short augmenting paths or cycles. Instead, it is based on linear programming and on a deterministic rounding scheme that was introduced in [25]. As a result, the matching computed by the algorithm of [2] does not have the nice structural properties of the matchings computed by algorithms based on the Hopkraft/Karp framework (such as not having any short augmenting paths or cycles).

Our randomized weighted matching algorithm. The general idea of our approach is as simple as the algorithm for the unweighted case in [49].[1] We first describe the randomized version of our algorithm. We start with an initial matching M_0 of the given weighted graph $G = (V, E, w)$ and it then consists of iterations $i = 1, 2, \ldots$. In each iteration, we update the given matching such that at the end of iteration i, we have matching M_i. In each iteration i, we sample a bipartite subgraph $H_i = (\hat{V}_i, \hat{E}_i)$ of G as follows. Every node $v \in V$ colors itself black or white independently with probability 1/2. An edge is called *monochromatic* if both of its endpoints have the same color, otherwise, we call the edge *bichromatic*. To preserve good intermediate results, we keep all the matching edges of the previous matching not occurring in the bipartition, i.e., we keep monochromatic matching edges. We call a node *free* regarding to matching M if none of its incident edges are $\in M$.

[1] Our algorithm is essentially the same one as the one in [7, 49]. We just replace the bipartite matching algorithm used as a subroutine. The analysis is then however different from the analysis in [7, 49].

Algorithm 2 Construct bipartite subgraph $\boldsymbol{H_i = (\hat{V}_i, \hat{E}_i)}$ of $\boldsymbol{G}$ based on matching $\boldsymbol{M_{i-1}}$.

1. Color each node black or white.
2. $\hat{V}_i := \{u \in V \mid u \text{ is free or } \exists \{u, v\} \in M_{i-1} \text{ s.t. } \{u, v\} \text{ is bichromatic}\}$.
3. $\hat{E}_i := \{\{u, v\} \in E \mid u, v \in \hat{V}_i \text{ and } \{u, v\} \text{ is bichromatic}\}$.

After sampling the bipartite subgraph H_i, we use the algorithm of [2] to compute a $(1-\lambda)$-approximate weighted matching of H_i for a sufficiently small parameter $\lambda > 0$ (λ will be exponentially small in $1/\varepsilon$). We then update the existing matching M_{i-1} by replacing the bichromatic matching edges with the matching edges of the newly computed matching of H_i. Because there is a collection of short augmenting paths and cycles with total gain $\Theta(w(M^*) - w(M_{i-1}))$ (where M^* is an optimal matching of G), we have a reasonable chance of sampling such paths and cycles so that the matching on H_i can potentially be improved by a sufficiently large amount. Note however that our algorithm does not guarantee that the quality of the matching improves monotonically during the algorithm. However, because the algorithm of [2] has deterministic guarantees if we choose λ sufficiently small, we have the guarantee that $w(M_i)$ cannot be much worse than $w(M_{i-1})$. With the right choice of parameters, it turns out that $2^{O(1/\varepsilon)}$ iterations are sufficient to obtain a $(1-\varepsilon)$-approximate matching with at least constant probability.

Our deterministic weighted matching algorithm. The basic idea of our deterministic algorithm is the same as for the randomized algorithm. However, we now have to compute the bipartition into black and white nodes in each iteration deterministically. For this purpose, we prove in Lemma 13 that for some $T = 2^{O(1/\varepsilon)} \cdot \ln n$, there exist a collection of bipartitions $H_1, \dots, H_T$ such that every path/cycle of length at most $O(1/\varepsilon)$ (and thus also every augmenting path/cycle of this length) of G appears in at least one of these bipartitions. Of course, after changing the matching, also the set of augmenting paths and cycles changes and one can therefore not just iterate over all bipartitions $H_1, \dots, H_T$, improve the matching for each bipartition by using a generic MWM approximation algorithm, and guarantee that at the end the resulting matching is a sufficiently good approximation. However, the property of $H_1, \dots, H_T$ guarantees that for a fixed initial matching M, when going over all T bipartitions, there exists one bipartition that can improve the weight of M by $\Theta(w(M)/T)$. We therefore proceed as follows. We iterate $O(T)$ times through the sequence $H_1, \dots, H_T$ of bipartitions. For each bipartition, we use the (deterministic) algorithm of [2] to compute a $(1-\lambda)$-approximate weighted matching of the current bipartite graph. We then however only switch to the new matching if it improves the old one by a sufficiently large amount. Checking if a given bipartition leads to a sufficiently large improvement of the current matching can be done efficiently by first applying the diameter reduction technique given by Theorem 7.

6 Vertex Cover Lower Bound

Göös and Suomela in [32] showed that there exists a constant $\varepsilon_0 > 0$ such that computing a $(1+\varepsilon_0)$-approximation of minimum (unweighted) vertex cover in bipartite graphs of maximum degree 3 requires $\Omega(\log n)$ rounds even in the LOCAL model. We next describe how to extend this lower bound to show that for $\varepsilon \leq \varepsilon_0$, computing a $(1+\varepsilon)$-approximate vertex cover for any $\varepsilon > 0$. That is, we next proof Theorem 5, i.e., we prove that even in bipartite graph of maximum degree 3, there exists a constant $\varepsilon_0 > 0$ such that for $\varepsilon \in (0, \varepsilon_0]$, computing a $(1+\varepsilon)$-approximate vertex cover requires $\Omega\big(\frac{\log n}{\varepsilon}\big)$ rounds in the LOCAL model.

Proof of Theorem 5. in [32], Göös and Suomela showed that there exists a bipartite graph $G = (V_G, E_G)$ with maximum degree 3 and a constant $\varepsilon_0 > 0$ such that no randomized distributed algorithm with running time $o(\log n)$ can find a $(1+\varepsilon_0)$-approximate vertex cover on G. To extend this proof to smaller approximation ratios, we proceed as follows. Given a positive integer parameter k, we construct a new lower bound graph H as follows. Graph H is obtained from graph G by replacing every edge e of G by a path P_e of length $2k+1$.

We first describe a method to transform a given vertex cover S_H of G into a vertex cover S'_H of H such that $|S'_H| \leq |S_H|$ and such that S'_H has the following form. For each edge e of G, S'_H contains exactly k of the inner nodes of the path P_e in H and it contains at least one of the end nodes of P_e. The transformation is done independently for each path P_e as follows. Let $P_e = (v_0, v_1, \ldots, v_{2k+1})$ be the path that replaces edge $e = \{v_0, v_{2k+1}\}$ in G. If $S_H \cap \{v_0, v_{2k+1}\} \neq \emptyset$, we add the nodes $S_H \cap \{v_0, v_{2k+1}\}$ also to S'_H. If $S_H \cap \{v_0, v_{2k+1}\} = \emptyset$, we arbitrarily add either v_0 or v_{2k+1} to S'_H. Further S'_H contains exactly k of the inner nodes $v_1, \ldots, v_{2k}$ of P_e in such a way that every edge of P_e is covered by some node in S'_H. If $v_0 \in S'_H$, we add all v_{2i} for $i \in \{1, \ldots, k\}$ and otherwise we add all v_{2i-1} for $i \in \{1, \ldots, k\}$. Clearly S'_H is a vertex cover of H. To see that $|S'_H| \leq |S_H|$, observe that in order to cover all $2k+1$ edges of P_e, every vertex cover of H must contain at least $k+1$ of the nodes of P_e and it also must contain at least k of the inner nodes of P_e. If $S_H \cap \{v_0, v_{2k+1}\} \neq \emptyset$, S'_H only differs in terms of the inner nodes of P_e from S_H and we know that also S_H must contain at least k inner nodes of P_e. If $S_H \cap \{v_0, v_{2k+1}\} = \emptyset$, we add either v_0 or v_{2k+1} to S'_H. However in this case, S_H contains at least $k+1$ inner nodes of P_e and S'_H only contains k inner nodes of P_e. The transformation from S_H to S'_H can be done independently for each of the paths P_e of length $2k+1$ and it can therefore be done in $O(k)$ rounds in the LOCAL model.

Let e_G be the number of edges of G and let s_G be the size of an optimal vertex cover of G. Because the maximum degree of G is 3 and we have an optimal vertex cover of G, we get that $e_G = c \cdot s_G$ for some constant $c \leq 3$. By the observation above, any vertex cover S_H of H can be transformed into an equally good vertex cover S'_H with a nice structure. Note that S'_H consists of exactly k inner nodes of each path P_e for $e \in E_G$ and it consists of at least one of the end nodes of each such path (i.e., of a vertex cover of G). We therefore obtain that there is an optimal vertex cover of H that consists of an optimal vertex cover of G and of k inner nodes of each $(2k+1)$-hop path P_e replacing an edge e of G. The size s_H of an optimal vertex cover of H is therefore exactly $s_H = s_G + k \cdot e_G = (1+ck) \cdot s_G$.

Assume now that we have a T-round algorithm to compute a $(1+\varepsilon)$-approximate vertex cover S_H on graph H for some $\varepsilon \leq \varepsilon_0/(1+3k) \leq \varepsilon_0/(1+ck)$. By the above observation, in $O(k)$ rounds, we can transform this vertex cover into a vertex cover S'_H, which contains $k \cdot e_G = ck \cdot s_G$ inner path nodes and at least one of the end nodes of each $(2k+1)$-hop path replacing an edge of G in H. The vertex cover S'_H of H therefore induces a vertex cover of G of size $|S'_H| - k \cdot e_G = |S'_H| - ck \cdot s_G$. Because we assumed that $\varepsilon \leq \varepsilon_0/(1+ck)$, we have $|S'_H| \leq (1+\varepsilon) \cdot (1+ck) \cdot s_G \leq (1+\varepsilon_0)s_G + ck \cdot s_G$. The vertex cover S'_H of H therefore induces a $(1+\varepsilon_0)$-approximate vertex cover of G. We next show that this implies an $O(1+T/k)$-round algorithm to compute a $(1+\varepsilon_0)$-approximate vertex cover of G. Assume that we want to compute a vertex cover of G. To do this, the nodes of G can simulate graph H by adding $2k$ virtual nodes on each edge of G. An R-round algorithm on H can then be run in $O(\lceil R/k \rceil) = O(1+R/k)$ rounds on G. We can therefore compute the vertex cover S'_H on the virtual graph H in $O(1+(T+k)/k) = O(1+T/k)$ rounds on G. Since $k = \Theta(1/\varepsilon)$, the lower bound of [32] implies a lower bound of $\Omega\big(\frac{\log n}{\varepsilon}\big)$ on the time T for computing a $(1+\varepsilon)$-approximate vertex cover on H. Finally note that since G is bipartite, then H is also bipartite and if the maximum degree of G is 3, then the maximum degree of H is also 3. ◀

References

1 M. Ahmadi and F. Kuhn. Distributed maximum matching verification in CONGEST. In *Proc. 34th Symp. on Distributed Computing (DISC)*, pages 37:1–37:18, 2020.

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

3 S. Assadi, M. Bateni, A. Bernstein, V. Mirrokni, and C. Stein. Coresets meet EDCS: Algorithms for matching and vertex cover on massive graphs. In *Proc. 30th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1616–1635, 2019.

4 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.

5 M. Åstrand and J. Suomela. Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks. In *Proc. 22nd ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 294–302, 2010.

6 B. Awerbuch and D. Peleg. Sparse partitions. In *Proc. 31st IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 503–513, 1990.

7 R. Bar-Yehuda, K. Censor-Hillel, M. Ghaffari, and G. Schwartzman. Distributed approximation of maximum independent set and maximum matching. In *Proc. 36th ACM Symp. on Principles of Distributed Computing (PODC),* full version: `arXiv:1708.00276`, pages 165–174, 2017.

8 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.

9 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 *Proc. 35th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 3–8, 2016.

10 R. Ben-Basat, G. Even, Kawarabayashi K, and G. Schwartzman. A deterministic distributed 2-approximation for weighted vertex cover in $O(\log N\log\Delta/\log^2\log\Delta)$ rounds. In *Proc. 25th Coll. on Structural Information and Comm. Complexity (SIROCCO)*, pages 226–236, 2018.

11 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.

12 Ran Ben-Basat, Ken ichi Kawarabayashi, and Gregory Schwartzman. Parameterized distributed algorithms. In *Proc. 33rd In. Symp. on Distributed Computing (DISC)*, pages 6:1–6:16, 2019.

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

14 A. Czygrinow and M. Hańćkowiak. Distributed algorithm for better approximation of the maximum matching. In *9th Annual Int. Computing and Combinatorics Conf. (COCOON)*, pages 242–251, 2003.

15 A. Czygrinow, M. Hańćkowiak, and E. Szymanska. A fast distributed algorithm for approximating the maximum matching. In *Proceedings of 12th Annual European Symp. on Algorithms (ESA)*, pages 252–263, 2004.

16 R. Diestel. *Graph Theory*, chapter 2.1. Springer, Berlin, 3rd edition, 2005.

17 I. Dinur and S. Safra. On the hardness of approximating vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005.

18 J. Edmonds. Maximum matching and a polyhedron with 0, l-vertices. *J. of Res. the Nat. Bureau of Standards*, 69 B:125–130, 1965.

19 J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17(3):449–467, 1965.

20 J. Egerváry. Matrixok kombinatorius tulajdonsáairól [On combinatorial properties of matrices]. *Matematikai és Fizikai Lapok (in Hungarian)*, 38(16–28), 1931.

21 G. Even, M. Medina, and D. Ron. Distributed maximum matching in bounded degree graphs. In *Proceedings of the 2015 Int. Conf. on Distributed Computing and Networking (ICDCN)*, pages 18:1–18:10, 2015.

22 S. Faour, M. Fuchs, and F. Kuhn. Distributed CONGEST approximation of weighted vertex covers and matchings. *CoRR*, abs/2111.10577, 2021. `arXiv:2111.10577`.

23 S. Faour and F. Kuhn. Approximating bipartite minimum vertex cover in the CONGEST model. In *Proc. 24th Conf. on Principles of Distr. Systems (OPODIS)*, pages 29:1–29:16, 2020.

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

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

26 M. Fischer, S. Mitrovic, and J. Uitto. Deterministic $(1+\varepsilon)$-approximate maximum matching with poly$(1/\varepsilon)$ passes in the semi-streaming model. *CoRR*, abs/2106.04179, 2021. `arXiv:2106.04179`.

27 K.-T. Foerster, J. H. Korhonen, J. Rybicki, and S. Schmid. Brief announcement: Does preprocessing help under congestion? In *Proc. 38th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 259–261, 2019.

28 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.

29 M. Ghaffari and F. Kuhn. Deterministic distributed vertex coloring: Simpler, faster, and without network decomposition. In *Proc. 62nd IEEE Symp. on Foundations of Computer Science (FOCS)*, 2021.

30 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.

31 M. Ghaffari, F. Kuhn, Y. Maus, and J. Uitto. Deterministic distributed edge-coloring with fewer colors. In *Proc. 50th ACM Symp. on Theory of Comp. (STOC)*, pages 418–430, 2018.

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

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

34 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.

35 D. G. Harris. Distributed local approximation algorithms for maximum matching in graphs and hypergraphs. *SIAM J. Computing*, 49(4):711–746, 2020.

36 J. Håstad. Some optimal inapproximability results. *J. of the ACM*, 48(4):798–859, 2001.

37 D. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, 1997.

38 J.H. Hoepman, S. Kutten, and Z. Lotker. Efficient distributed weighted matchings on trees. In *Proc 13th Coll. on Structural Inf. and Comm. Complexity (SIROCCO)*, pages 115–129, 2006.

39 J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

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

41 G. Karakostas. A better approximation ratio for the vertex cover problem. *ACM Trans. Algorithms*, 5(4):41:1–41:8, 2009.

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

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

44 S. Khuller, U. Vishkin, and N. E. Young. A primal-dual parallel approximation technique applied to weighted set and vertex covers. *J. Algorithms*, 17(2):280–289, 1994.

45 C. Koufogiannakis and N. E. Young. Distributed algorithms for covering, packing and maximum weighted matching. *Distributed Computing*, 24(1):45–63, 2011.

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

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

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

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

50 Z. Lotker, B. Patt-Shamir, and A. Rosén. Distributed approximate matching. *SIAM Journal on Computing*, 39(2):445–460, 2009.

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

52 T. Nieberg. Local, distributed weighted matching on general and wireless topologies. In *Proc. Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 87–92, 2008.

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

54 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.

55 S. Schmid and J. Suomela. Exploiting locality in distributed SDN control. In *Proc. 2nd ACM SIGCOMM Works. on Hot Topics in Software Defined Netw. (HotSDN)*, pages 121–126, 2013.

56 M. Wattenhofer and R. Wattenhofer. Distributed weighted matching. In *Proc. 18th Conf. on Distributed Computing (DISC)*, pages 335–348, 2004.

A Basic Tools

A.1 Fractional Approximation Algorithm

In all our upper bounds, we need an efficient deterministic distributed approximation scheme for the fractional variants of the MWVC and the MWM problem. In [2], Ahmadi, Kuhn, and Oshman showed that for every instance of the MWM problem with weights in the range $[1, W]$ and for every $\varepsilon \in (0, 1]$, it is possible to deterministically compute a $(1-\varepsilon)$-approximate fractional solution in time $O(\log(\Delta W)/\varepsilon^2)$ in the CONGEST model. In our algorithm to solve the MWVC problem, we need a variant of this algorithm, which works for the (unweighted) fractional w-matching problem. The following theorem shows that this can be done with the same asymptotic cost that we have for the fractional MWM problem.

▶ **Theorem 12.** *Let $G = (V, E, w)$ be an undirected n-node graph with integer node weights $w(v) \in \{1, \dots, W\}$. Then, for every $\varepsilon \in (0, 1]$, there is a deterministic $O(\log(\Delta W)/\varepsilon^2)$-round* CONGEST *algorithm to compute a $(1-\varepsilon)$-approximate solution to the fractional w-matching problem in G and a $(1+\varepsilon)$-approximate solution to the minimum fractional weighted vertex cover problem.*

Proof. The theorem could be proven for arbitrary weights in the range $[1, W]$ by adapting the algorithm of [2]. We here give a generic reduction, which works for integer weights and which allows to use the result of [2] (almost) in a blackbox manner.

We define an unweighted graph $G' = (V', E')$ as follows. For every node $v \in V$, V' contains $w(v)$ nodes $(v, 1), \dots, (v, w(v))$. Further, for every edge $\{u, v\} \in E$, we add a complete bipartite graph between the corresponding nodes to G', that is, E' contains all edges $\{(u, i), (v, j)\}$ for $i \in \{1, \dots, w(v)\}$ and $j \in \{1, \dots, w(u)\}$. We then apply the unweighted fractional matching algorithm of [2] to compute a $(1-\varepsilon)$-approximate fractional matching of G'. The maximum degree of any node in G' is at most $\Delta \cdot W$ and when running the algorithm in the CONGEST model on G', the round complexity of the algorithm is therefore $O(\log(\Delta W)/\varepsilon^2)$ as claimed (cf. Theorem 2 in [2]).

For an edge $e \in E'$ of G', assume that z_e is the fractional matching value of e in the computed fractional matching of G'. We can transform the fractional matching of G' into a fractional w-matching of G as follows. For each edge $\{u,v\} \in E$ of G, we define $y_{\{u,v\}} := \sum_{i=1}^{w(u)} \sum_{j=1}^{w(v)} z_{\{(u,i),(v,j)\}}$. Note that we obtain a valid fractional w-matching because for every node $u \in V$ of G, the sum of the fractional values of its edges is at most equal to number of copies of u in G', which is equal to $w(u)$. In the other direction, given a fractional w-matching y_e of G, we can compute a fractional matching $z_{e'}$ of G' of the same size in the following way. For each edge $\{u,v\} \in E$ of G, we assign $z_{\{(u,i),(v,j)\}} := y_{\{u,v\}}/(w(u) \cdot w(v))$. The size of a maximum fractional w-matching on G is therefore equal to the size of a maximum fractional matching on G' and given a $(1-\varepsilon)$-approximation of maximum fractional matching on G', we therefore also obtain a $(1-\varepsilon)$-approximation of maximum fractional w-matching on G.

It remains to show that we can efficiently run the fractional matching algorithm of [2] in the CONGEST model on G. Since every node of G has potentially a large number of copies in G', it is not true that any CONGEST algorithm on G' can be run efficiently in the CONGEST model on G. However, the behavior of the algorithm of [2] is independent of the node IDs and since the algorithm is deterministic, all copies of a node $u \in V$ are symmetric and therefore behave in exactly the same way. Each node of G can therefore simulate all its copies in G' at no additional cost.

We note that the same reduction has also been used in [33], where a maximal matching of G' is used to compute a 2-approximate weighted vertex cover of G. ◀

A.2 Bipartitions to Cover all Paths

The following lemma is used as a tool for our deterministic maximum weighted matching algorithms. It shows that there exists a sequence of bipartitions that is "good" in every graph and for every collection of short augmenting paths and cycles.

▶ **Lemma 13.** *Let $N > 0$ and $k \leq N$ be two integers. There exists a collection of $T = O(k \cdot 2^k \cdot \log N)$ functions $f_1, \ldots, f_T \in [N] \to \{0,1\}$ such that for every vector $(x_1, \ldots, x_k) \in [N]^k$ with pairwise disjoint entries, there exists a function f_i in the collection such that $f_i(x_j) = j \bmod 2$ for every $1 \leq j \leq k$.*

Proof. We will prove the lemma with a probabilistic argument. The probability that a randomly chosen f_i maps some x_j to 0 respectively 1 is $1/2$. We say an function f_i *takes care* of $(x_1, \ldots, x_k)$ if $(f_i(x_1), \ldots, f_i(x_k)) \in (0,1,0,1,\ldots)$. The probability that f_i takes care for a specific vector is 2^{-k}. Further, the probability that no function in a collection of T random function takes care of a fixed vector is $(1-2^{-k})^T \leq e^{-T/2^k}$. Since there are N^k different vectors, the probability that there exits a vector such that no function f_i takes care of it, is at most $N^k \cdot e^{-T/2^k}$. Choosing $T > k \cdot 2^k \cdot \ln N$ pushes this probability below 1, which shows that there must exist a collection of T functions $f_1, \ldots, f_T$ s.t. for every possible vector at least one of those functions will take care. ◀